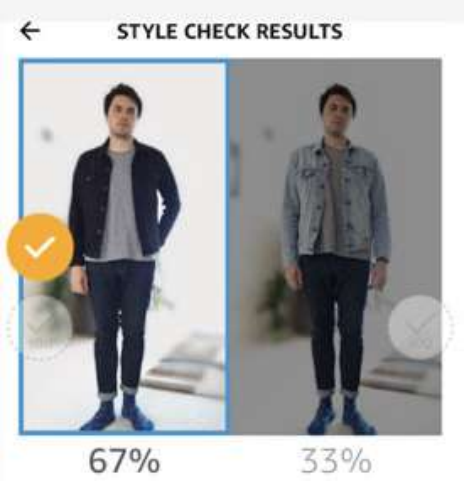
Internet of Things

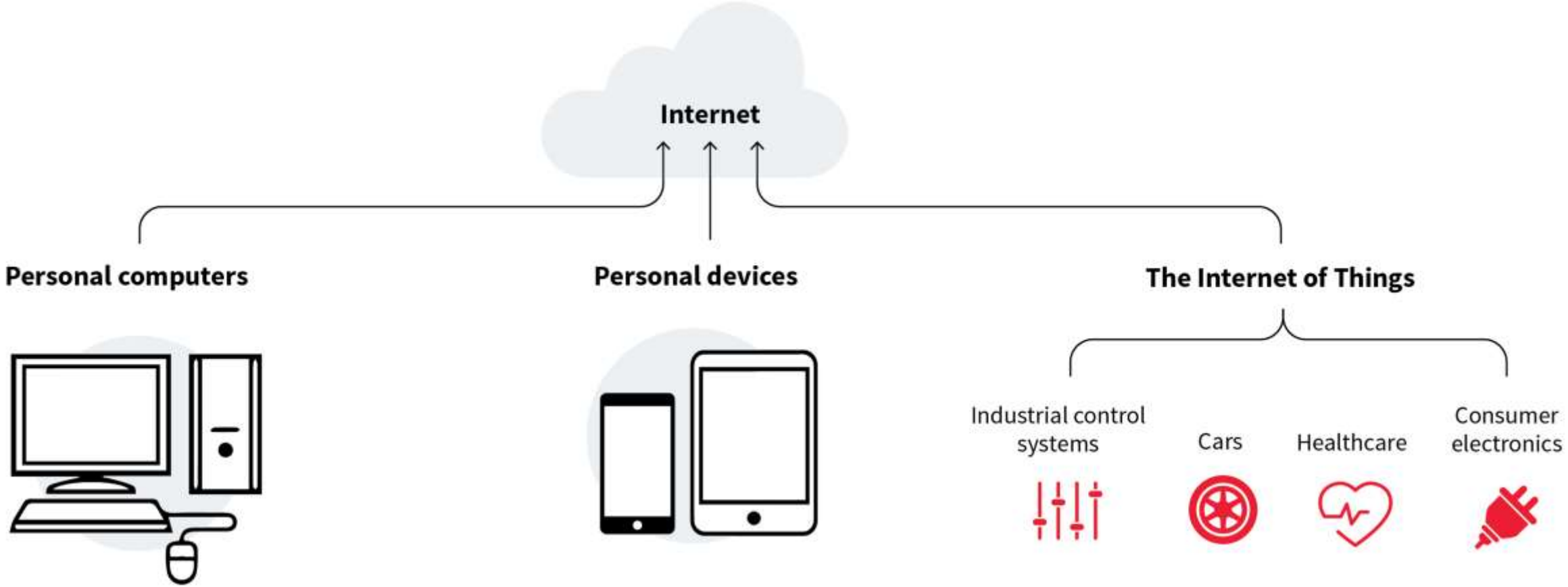
Internet of Things and NodeMCU

IoT Team, BFCAI

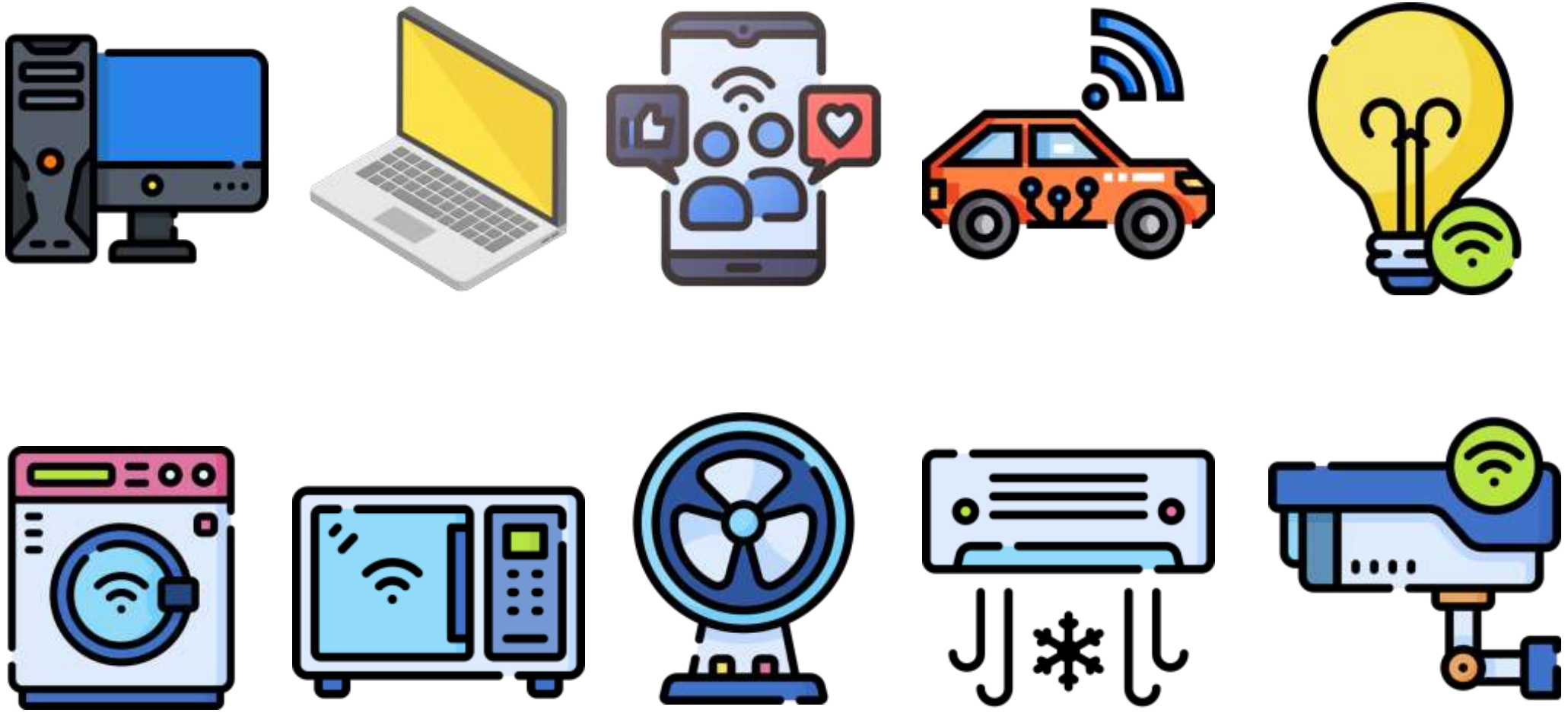
IoT Applications



Things


Internet
Personal computers
Personal devices
The Internet of Things
Industrial control systems
Cars
Healthcare
Consumer electronics

Things



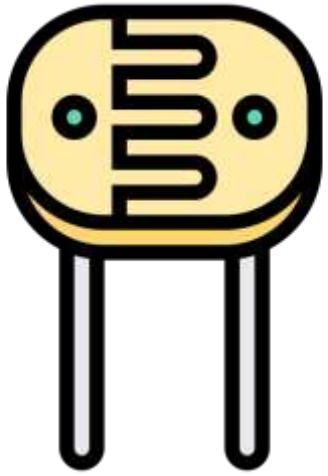
Internet of Things

- The Internet of Things (IoT) represents the network of physical objects "Things" that are integrated with sensors, software and other technologies for the purpose of exchanging data with other devices on the Internet.

Sensors

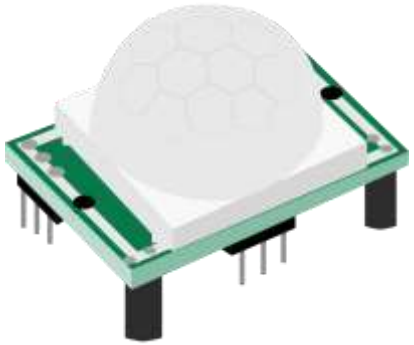
- A sensor is a device that detects some type of input from the physical environment.
- The input can be light, heat, motion, pressure or any number of other environmental phenomena.



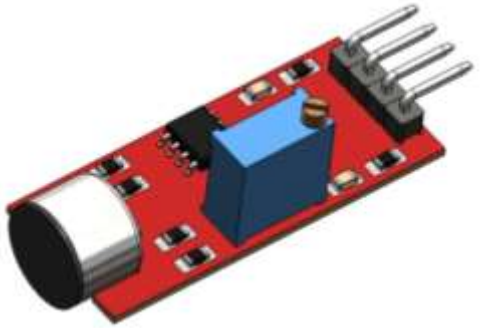



Sensors

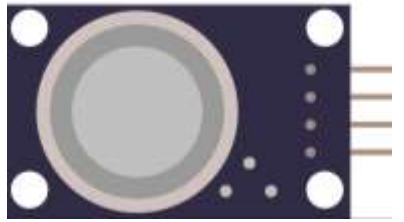
Temperature and Humidity



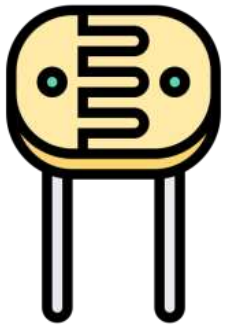
PIR Motion Detection



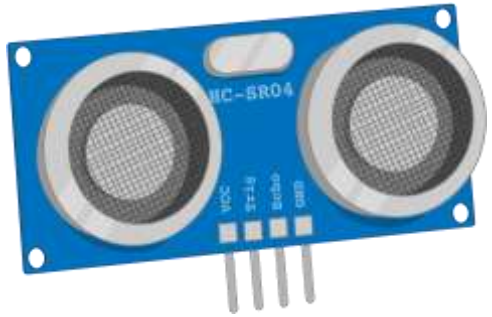
Microphone Sound Detection



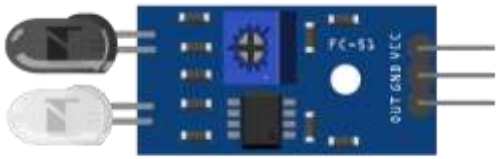
Gas/Smoke Sensor



Photoresistor CdS Sensor



Ultrasonic Sensor



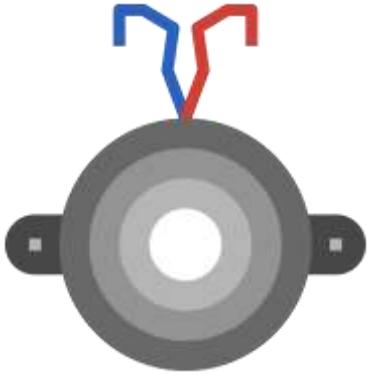
IR Obstacle Avoidance

Heart Rate Sensor (ECG)

Actuators

- Sensors turn a physical input into an electrical output, while actuators do the opposite.
- Actuators take electrical signals from control modules and turn them into physical outputs.

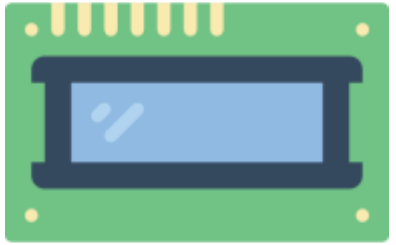
LEDs



Buzzer

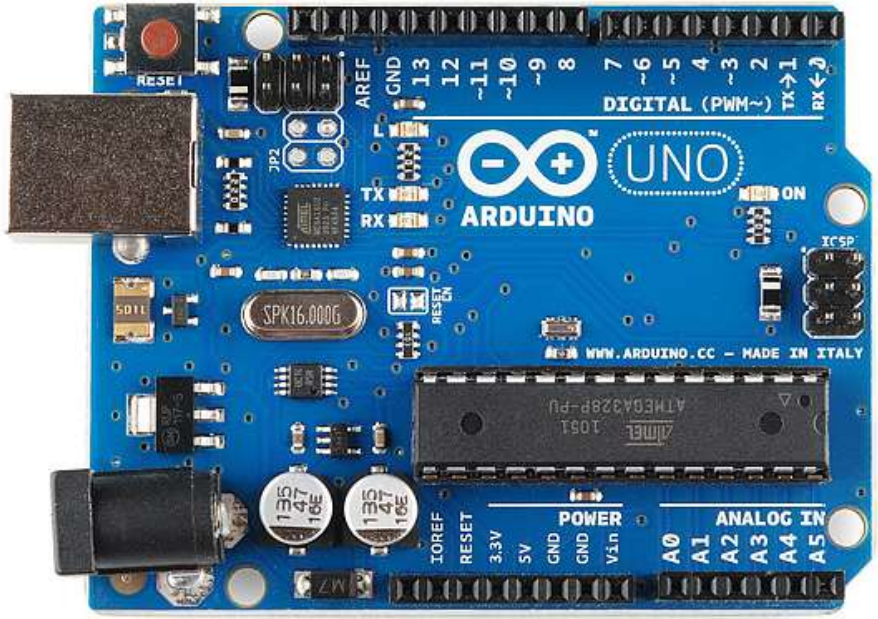
DC Fan

Servo Motor



LCD

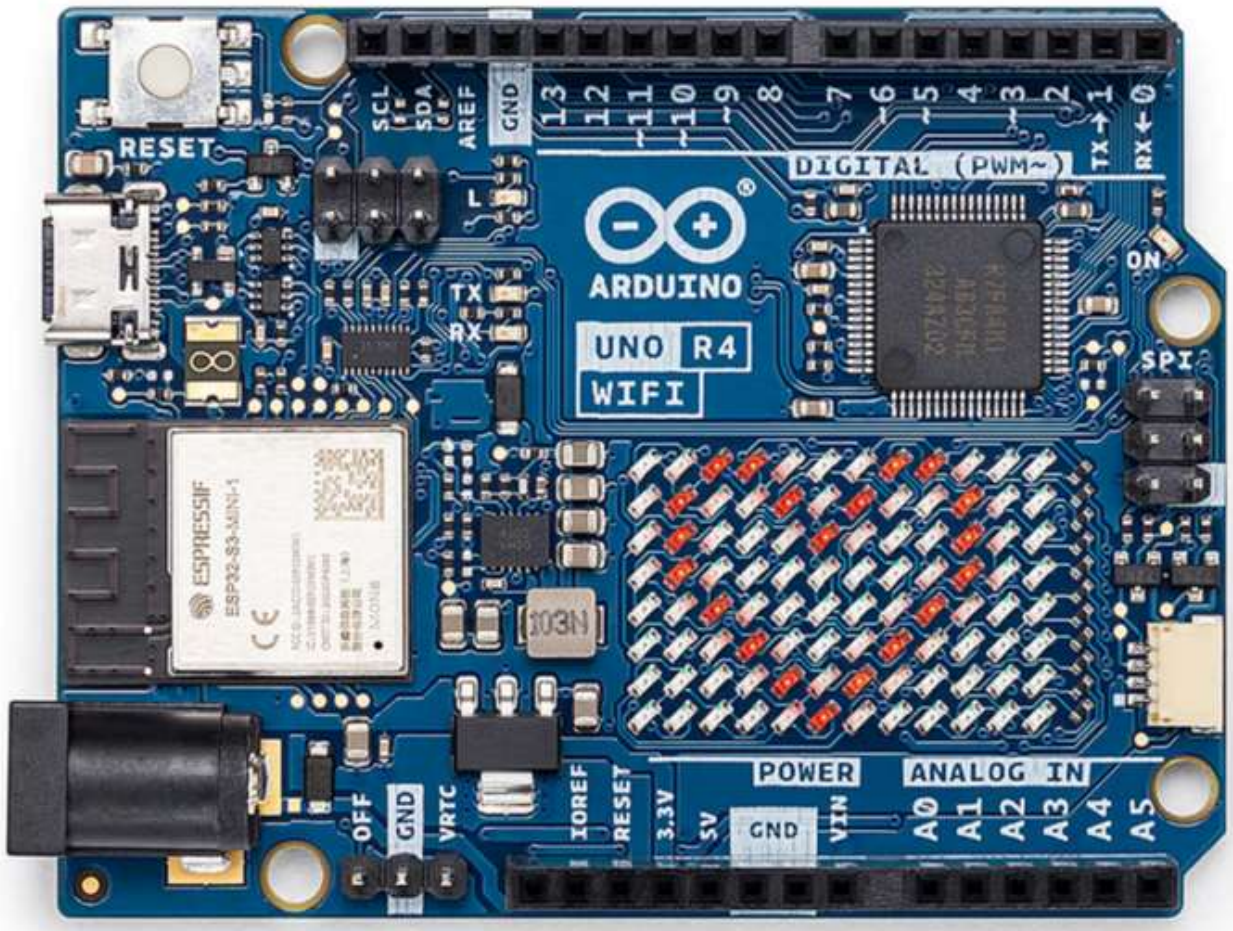
Processing – Arduino



Arduino Uno

Arduino Mega

Processing – Arduino R4 Wi-Fi



Arduino UNO R4 WIFI (China Version)

In Stock

EGP1,400.00

– 1 + Add to cart Add to Wishlist

Add to wishlist

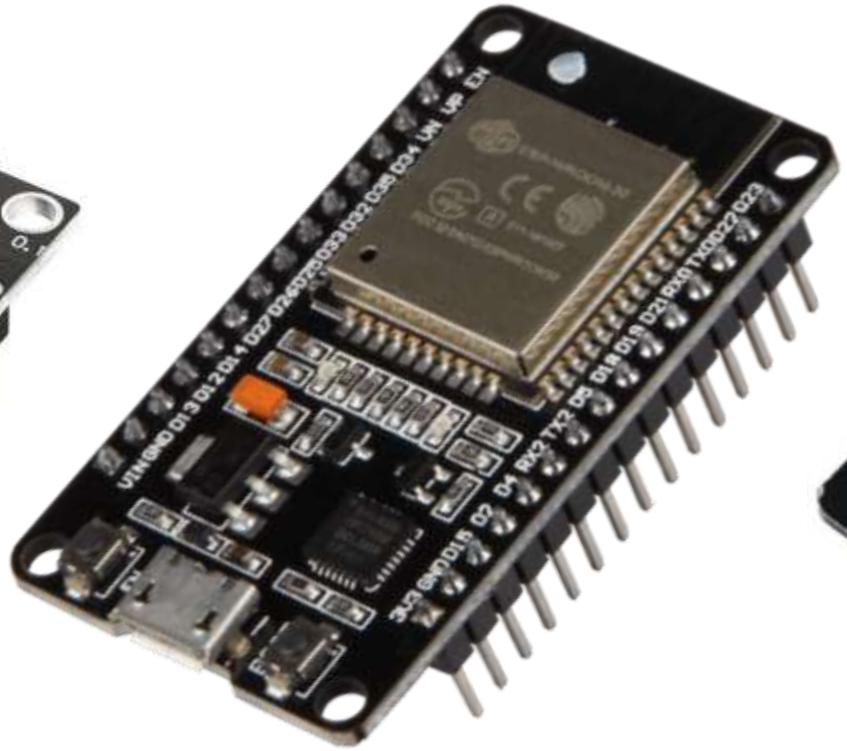
Categories: Arduino, Arduino Boards

Processing – Raspberry Pi

Raspberry Pi

Processing – ESP8266, ESP32 & ESP32-CAM

NodeMCU ESP8266



ESP32

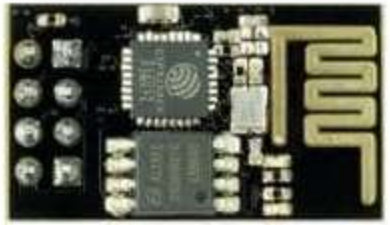
ESP32-CAM

NodeMCU ESP8266 or ESP32?

- The ESP8266 and the ESP32 are both low-cost microchips with Wi-Fi and microcontroller capabilities, making them well-suited for IoT applications.
- **Performance**: The ESP32 has a more powerful processor and more RAM than the ESP8266.
- **Connectivity**: The ESP32 supports Bluetooth in addition to Wi-Fi, which makes it better for applications that require both wireless technologies.
- **Power consumption**: The ESP32 has a lower power consumption than the ESP8266.
- **Price:** The ESP8266 (270 EGP) is cheaper than the ESP32 (430 EGP), making it a more budget-friendly option for projects.

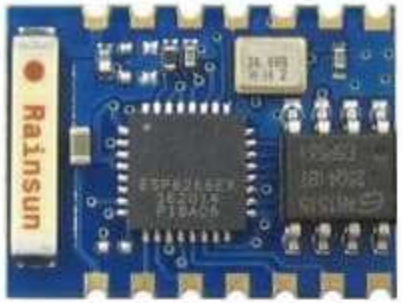
Different Types of Wi-Fi Modules

ESP-01



ESP-02

ESP-03



ESP-04

ESP-05

ESP-06

ESP-07

ESP-08

ESP-09

ESP-10

ESP-11

ESP-12

ESP-13

ESP-14

ESP-12 Versions

ESP-12E

ESP-12S

ESP-12F

ESP-12 Wi-Fi Module

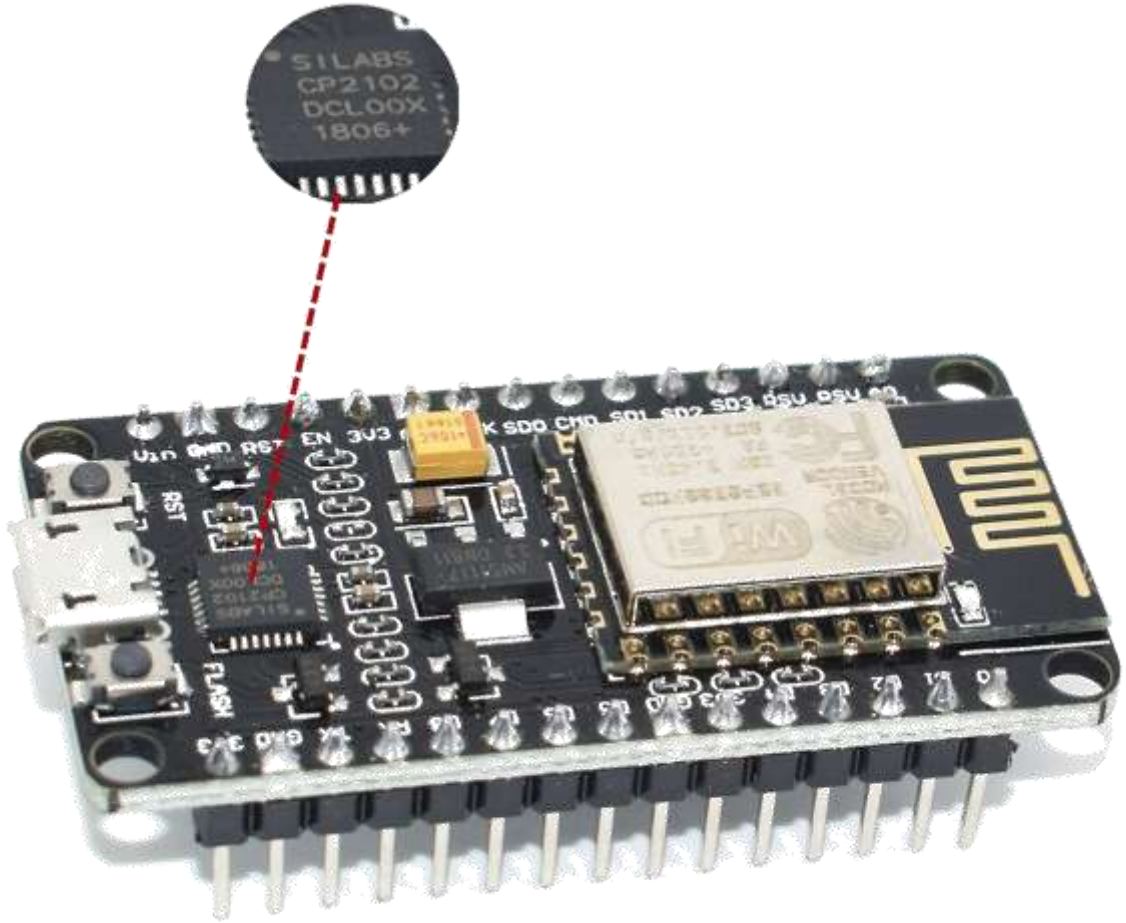
- ESP8266 is a cost-effective, easy-to-operate, compact-sized and low-powered Wi-Fi module.
- It supports both TCP/IP and Serial Protocols.
- It's normally used in IoT-based embedded projects and is considered the most widely used Wi-Fi module because of its low cost and small size.

NodeMCU ESP8266: Versions

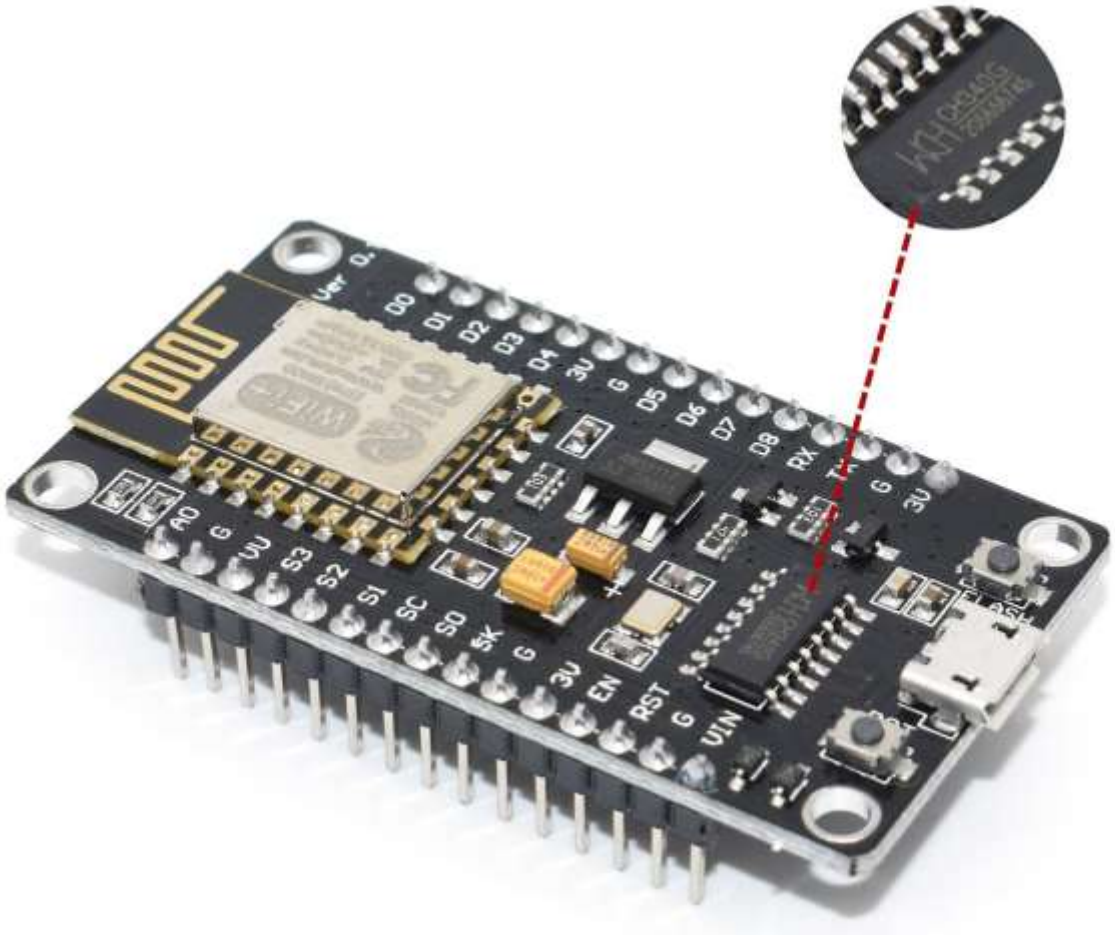
ESP8266 CP2102

ESP8266 CH340 V3

NodeMCU ESP8266: Versions



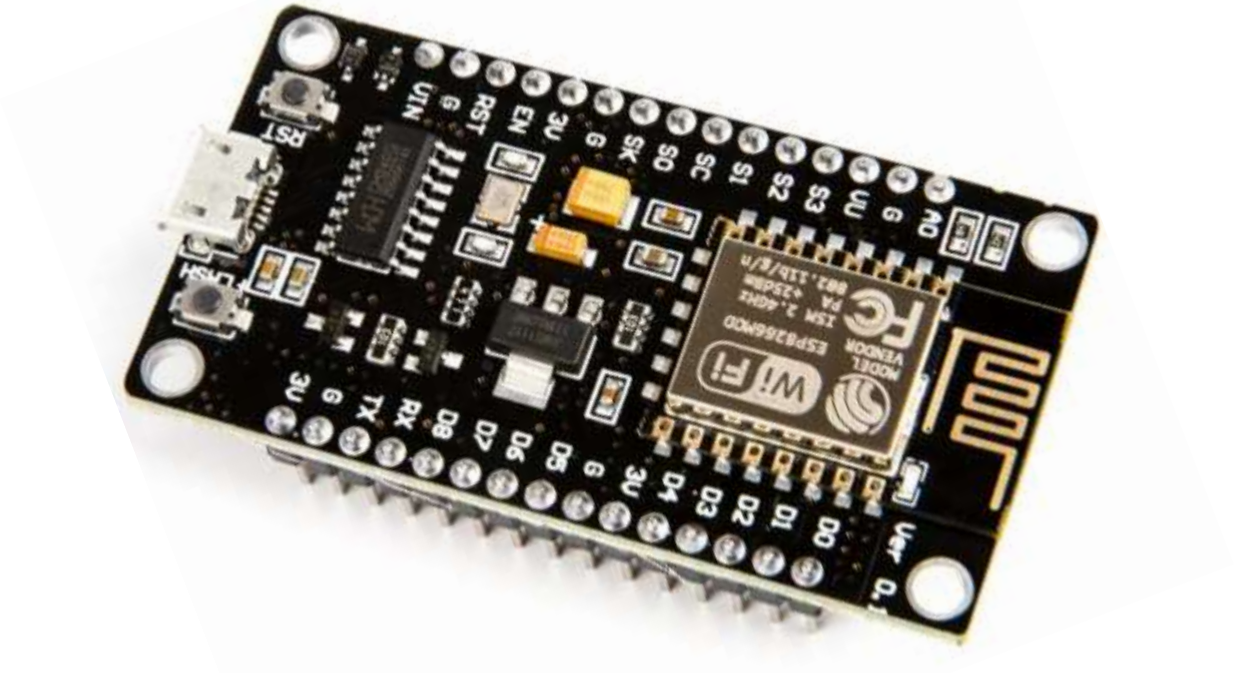
ESP8266 CP2102



ESP8266 CH340 V3

NodeMCU ESP8266 V3

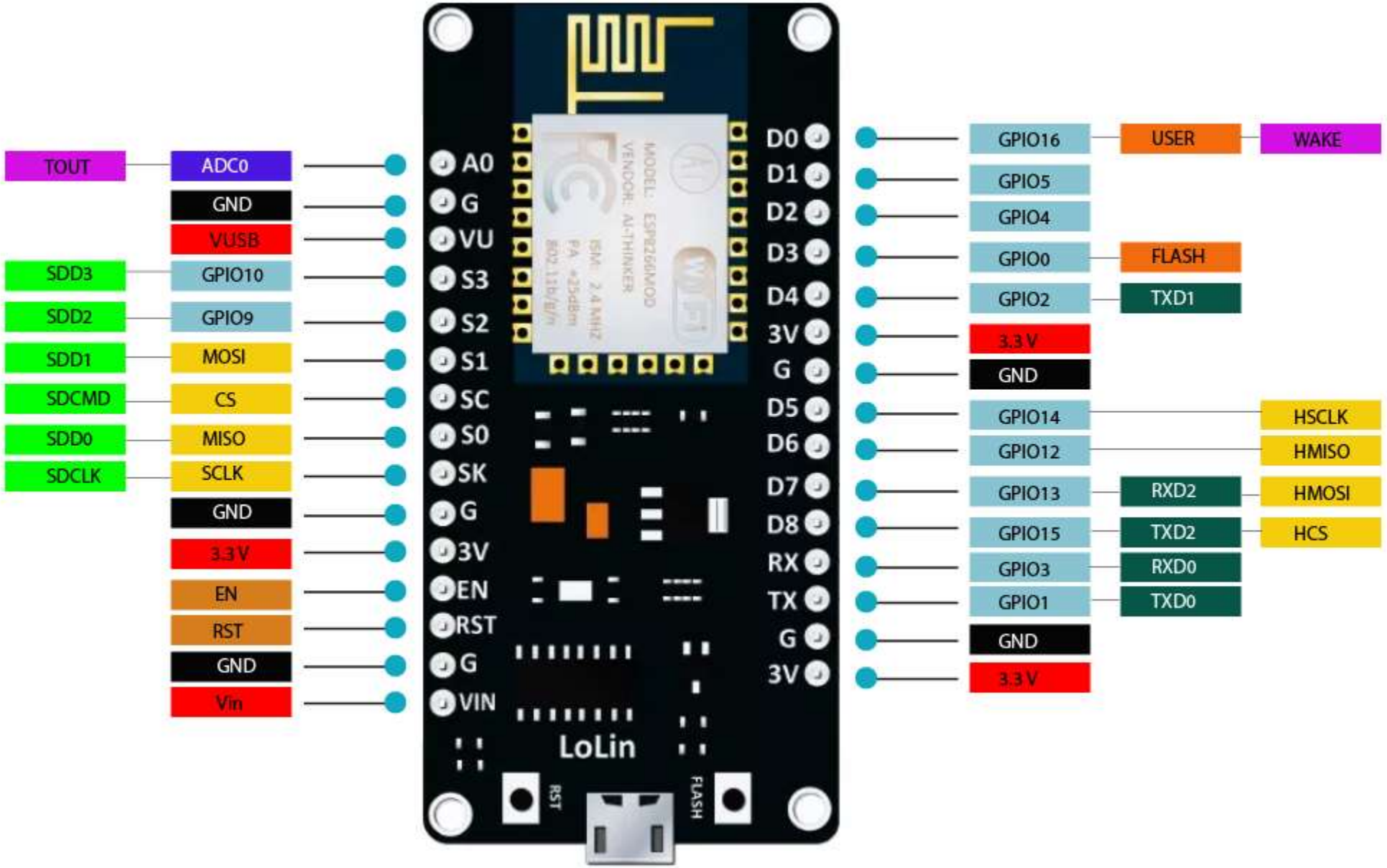
- NodeMCU is a low-cost open-source IoT platform based on the ESP8266 Wi-Fi system on a chip.
- NodeMCU Version 3 runs on the ESP-12E (ESP8266MOD) module, and it is easy-to-use development board equipped with analog and digital pins, a USB-to-serial adapter based on CH340G module, and a micro-USB.



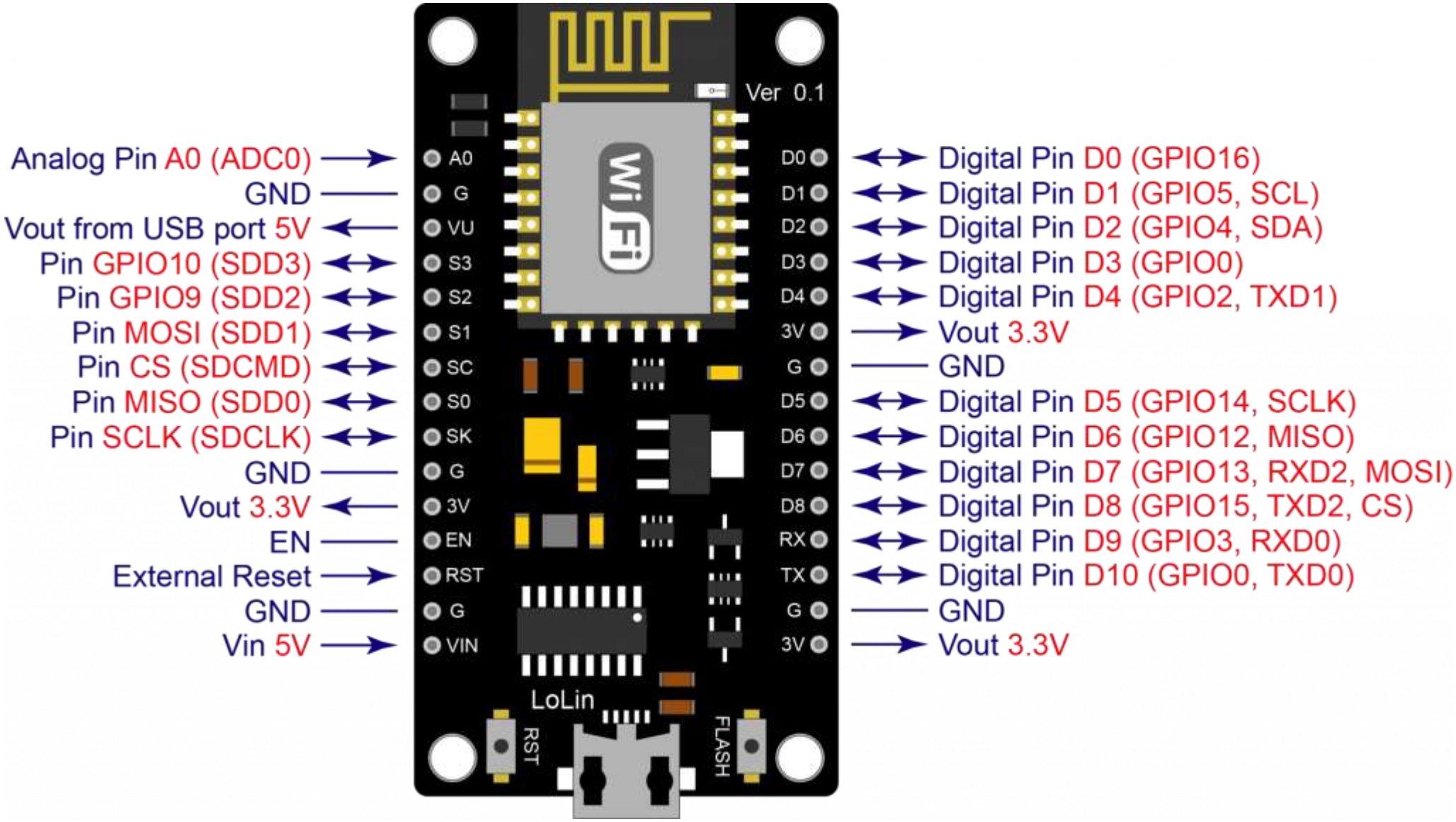
NodeMCU ESP8266 V3: Features

- Open-source
- Low Cost
- Arduino-like Hardware
- ESP8266 with Built-in Wi-Fi
- Programmable
- GPIO Pins
- PWM
- Status LED
- MicroUSB Port
- USB-to-UART Converter

NodeMCU ESP8266 V3: Pinout



NodeMCU ESP8266 V3: Pinout



NodeMCU ESP8266 V3: Specifications

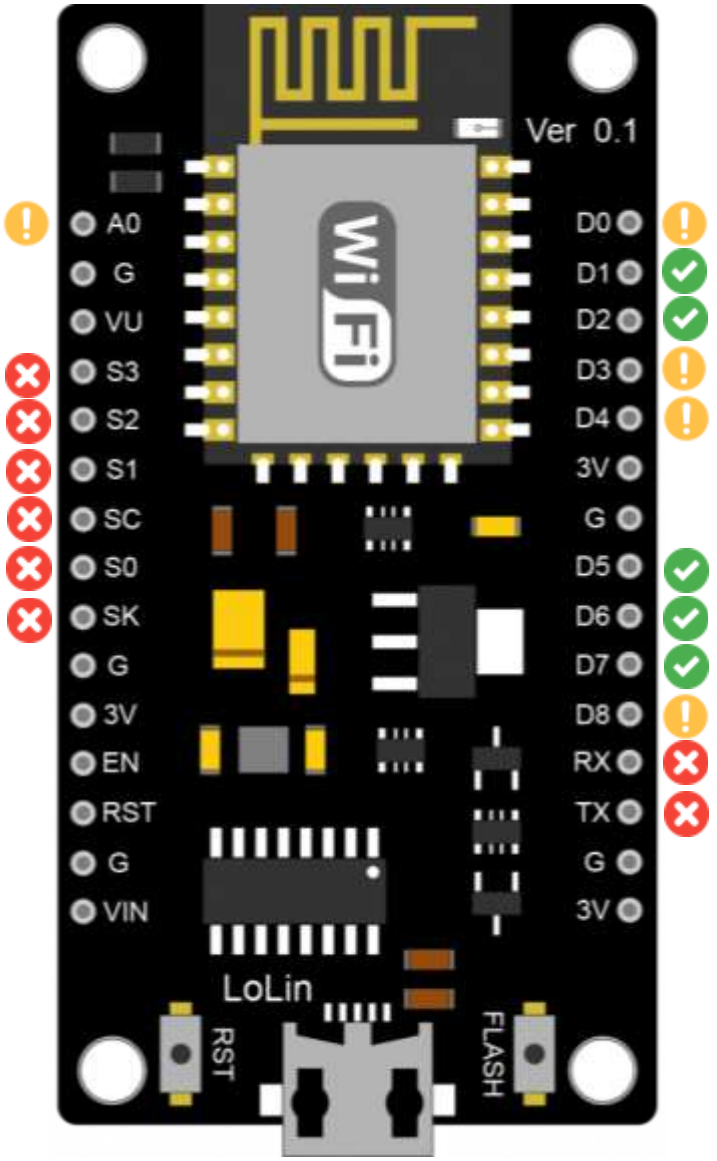
Microcontroller	L106 32-bit RISC
Clock Speed	80 MHz
Operating Voltage	3.3V
Digital I/O Pins	11
Analog Input Pins (ADC)	1
ADC Range	0-3.3V (10-bit)
PWM Resolution	0-1023 (10-bit)
Flash Memory	4 MB
SRAM	64 KB
USB-to-Serial	CH340G

NodeMCU ESP8266 V3: Specifications

Model	ESP8266-12E
Wireless Standard	802.11 b/g/n
Frequency Range	2.4 GHz
Wi-Fi Mode	Station / AP / AP+Station
Stack	Integrated TCP/IP
Data Interface	UART / I2C / HSPI / I2S
Encryption	WEP / TKIP / AES
Built-In WiFi	Yes
Built-In Bluetooth	No
USB Connector	Micro USB

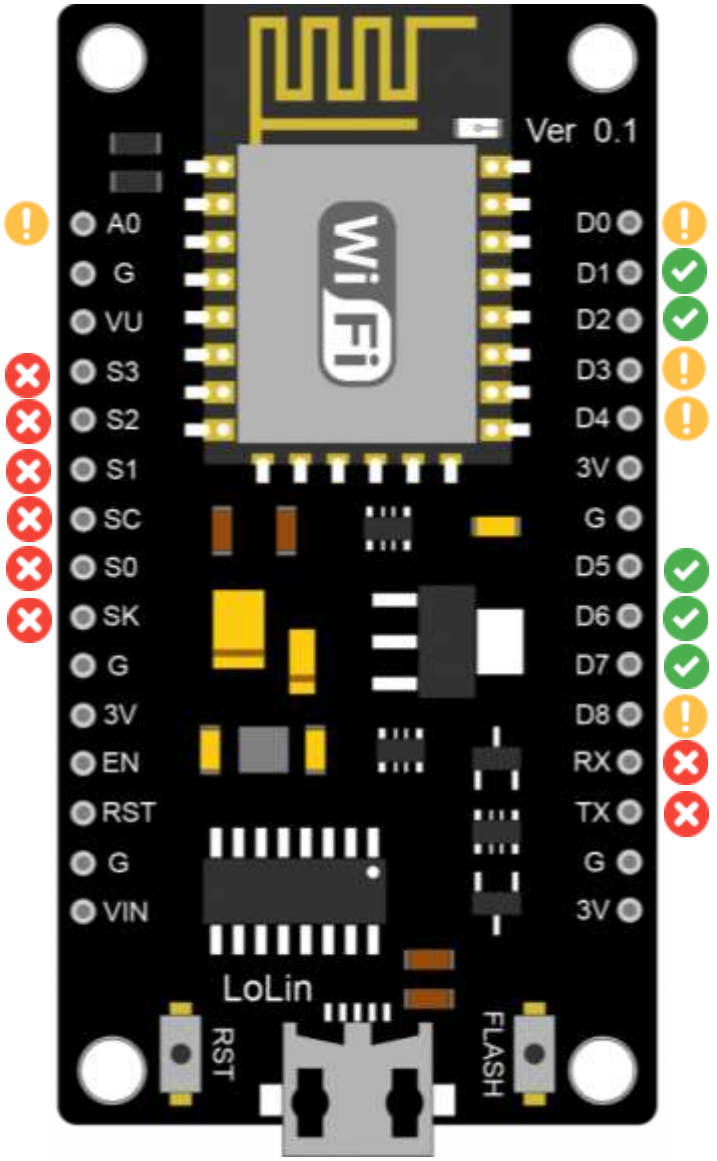
NodeMCU ESP8266 V3: Digital and Analog Pins

PIN	GPIO	Type
D0	GPIO16	Digital
D1	GPIO5	Digital
D2	GPIO4	Digital
D3	GPIO0	Digital
D4	GPIO2	Digital
D5	GPIO14	Digital
D6	GPIO12	Digital
D7	GPIO13	Digital
D8	GPIO15	Digital
A0	ADC0	Analog



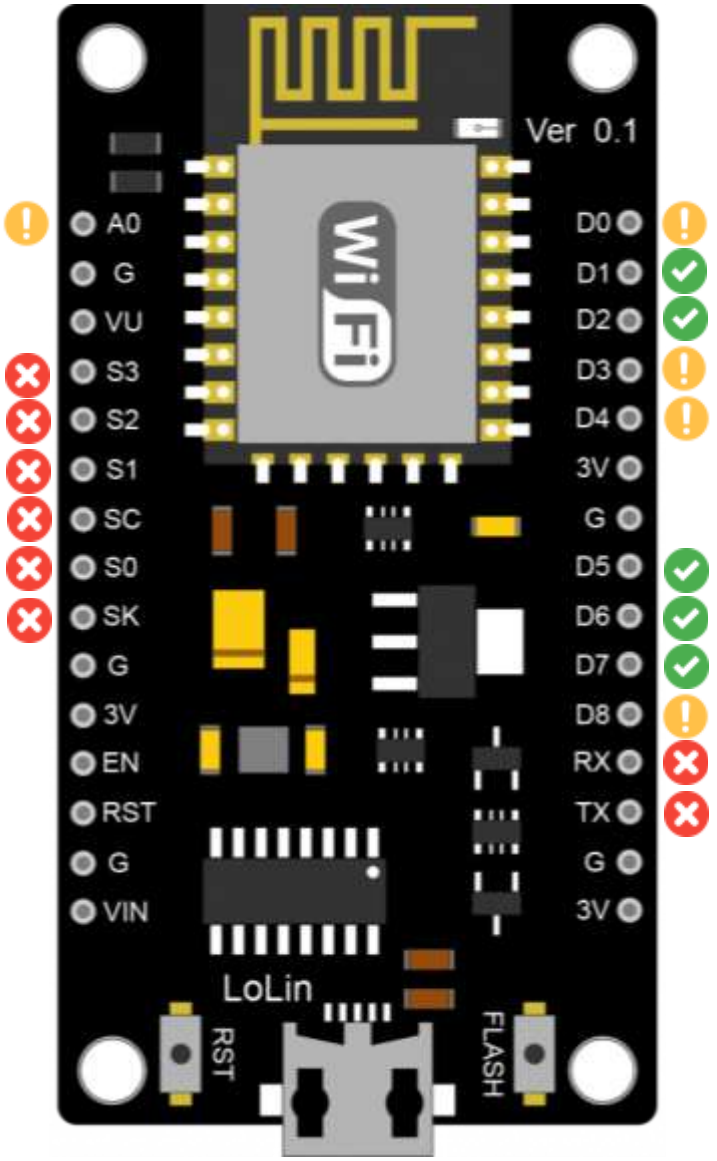
NodeMCU ESP8266 V3: GPIOs Safe to Use

PIN	GPIO	Why Not Safe?
D0	GPIO16	HIGH at boot Used to wake up from deep sleep
D1	GPIO5	-
D2	GPIO4	-
D3	GPIO0	Connected to FLASH button Boot fails if pulled LOW
D4	GPIO2	HIGH at boot Boot fails if pulled LOW
D5	GPIO14	-
D6	GPIO12	-
D7	GPIO13	-
D8	GPIO15	Required for boot Boot fails if pulled HIGH



NodeMCU ESP8266 V3: GPIOs Safe to Use

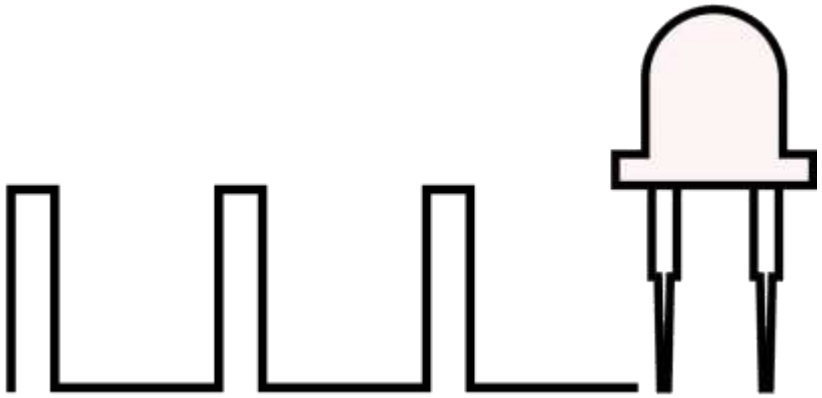
PIN	GPIO	Why Not Safe?
RX	GPIO3	Used for flashing and debugging
TX	GPIO1	Used for flashing and debugging
CLK	GPIO6	Connected to Flash memory
SDO	GPIO7	Connected to Flash memory
CMD	GPIO11	Connected to Flash memory
SD1	GPIO8	Connected to Flash memory
SD2	GPIO9	Connected to Flash memory
SD3	GPIO10	Connected to Flash memory
A0	ADC0	Analog input pin Cannot be configured as output

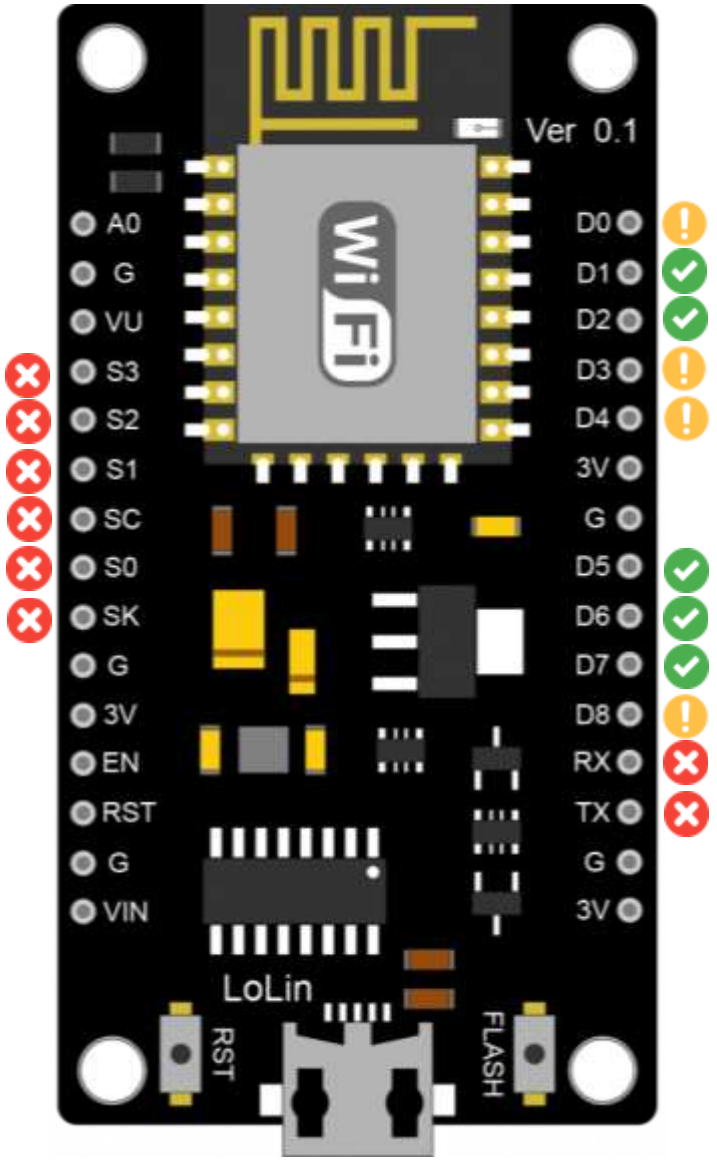


NodeMCU ESP8266 V3: PWM Pins

- All of the ESP8266's GPIO pins, from GPIO0 to GPIO15, can be programmed to generate pulse width modulated (PWM) outputs.
- On the ESP8266, the PWM signal has a 10-bit resolution.

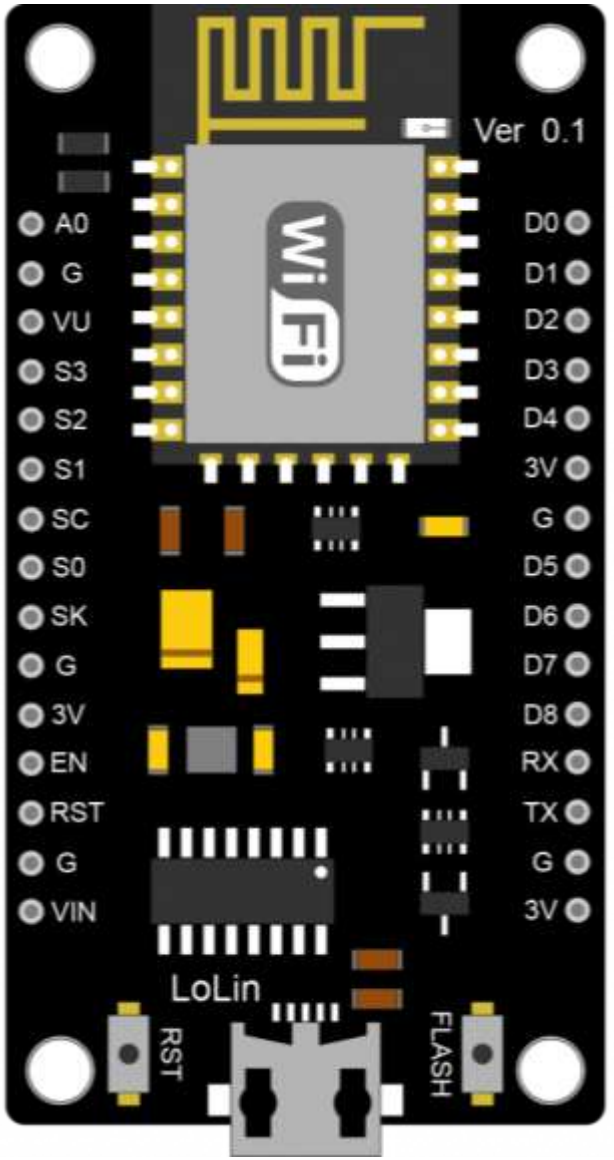
```
analogWrite(pin, 0);
analogWrite(pin, 1023);
```





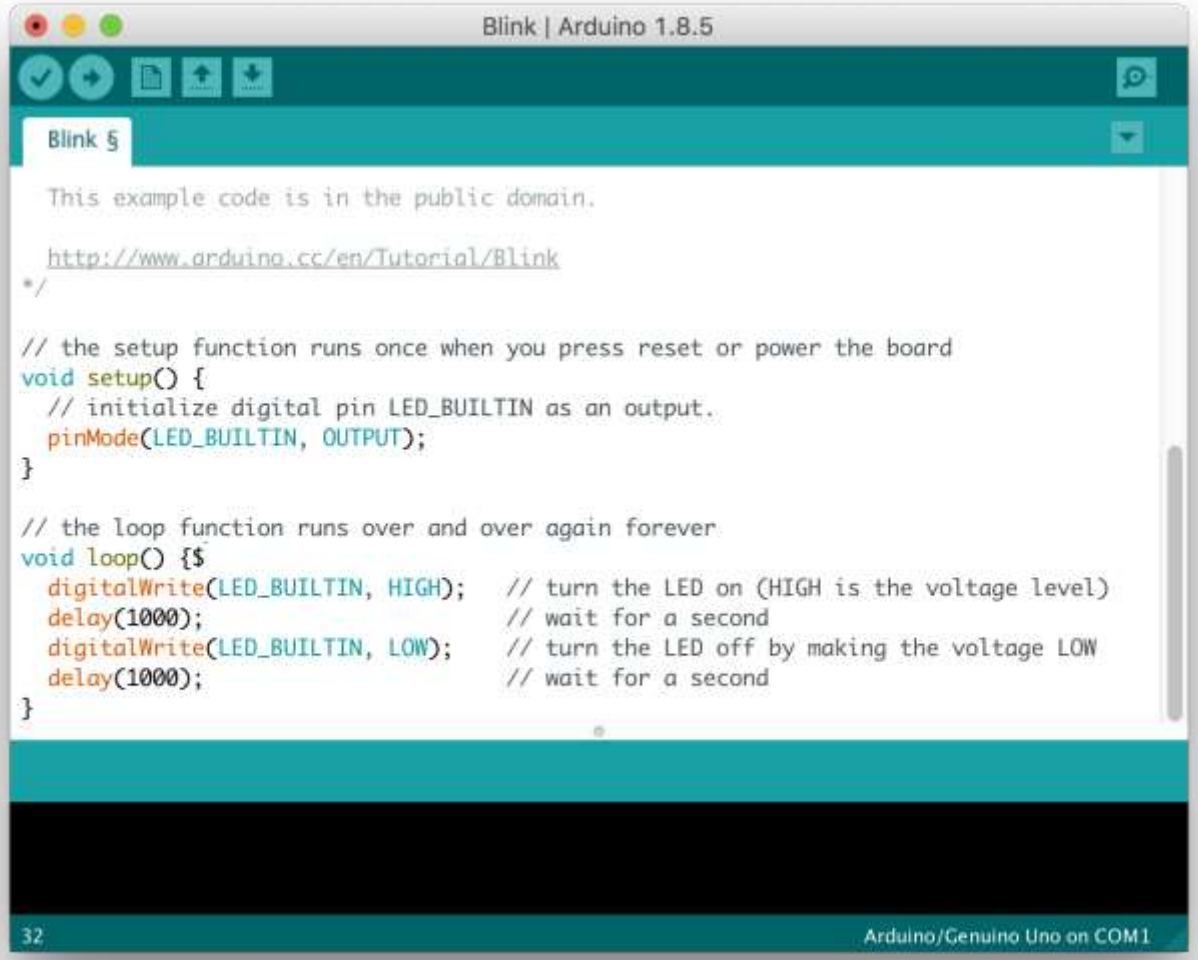
NodeMCU ESP8266 V3: Power Pins

PIN	Type
G	Ground Pin
3.3V	3.3V Power Pin
VIN	Can be Used to Directly Power the NodeMCU/ESP8266
VU	Voltage Supplied by the USB



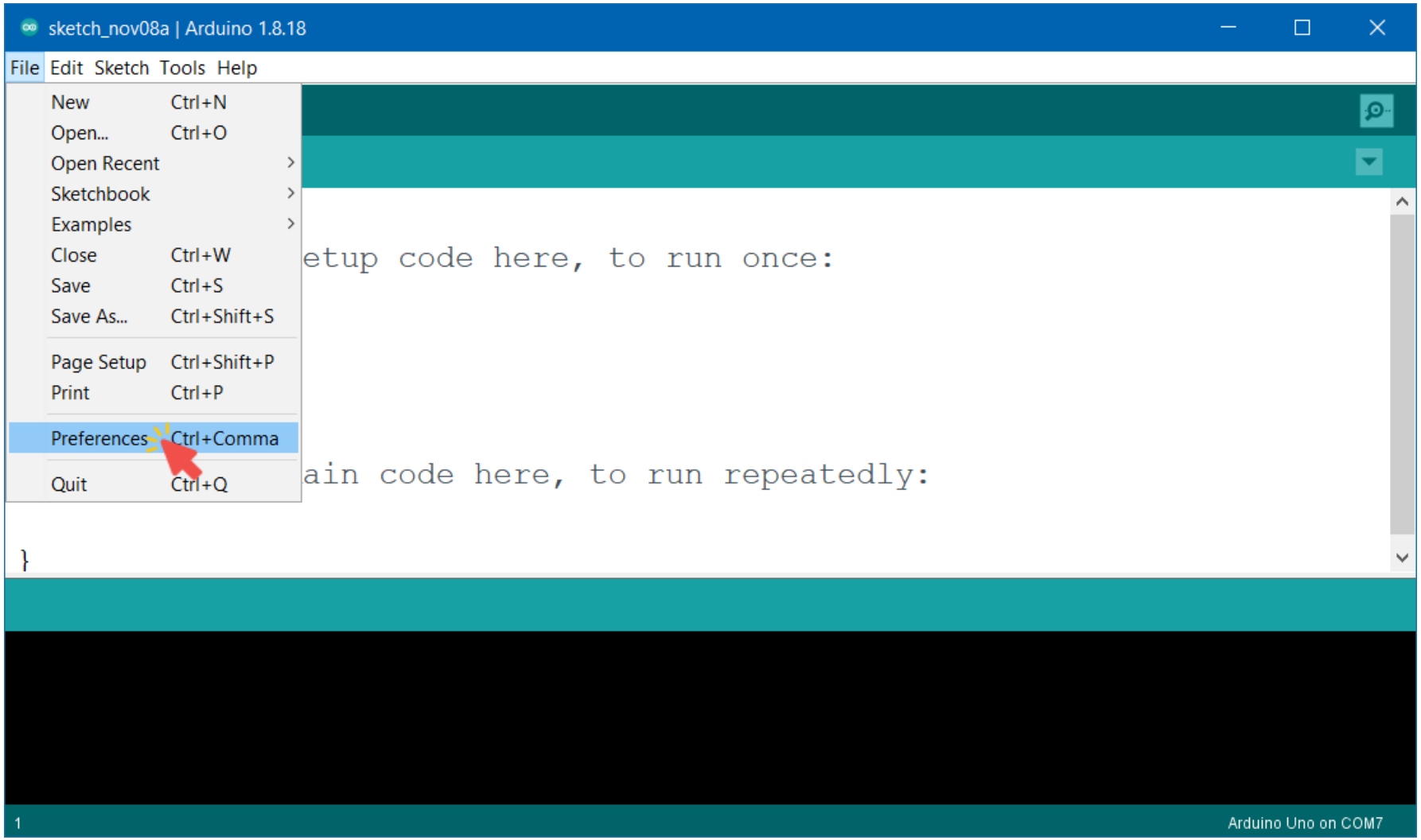
Arduino IDE

- The Arduino IDE enables you to write and edit code and convert this code into instructions that NodeMCU hardware understands.



Installing NodeMCU ESP8266 in Arduino IDE

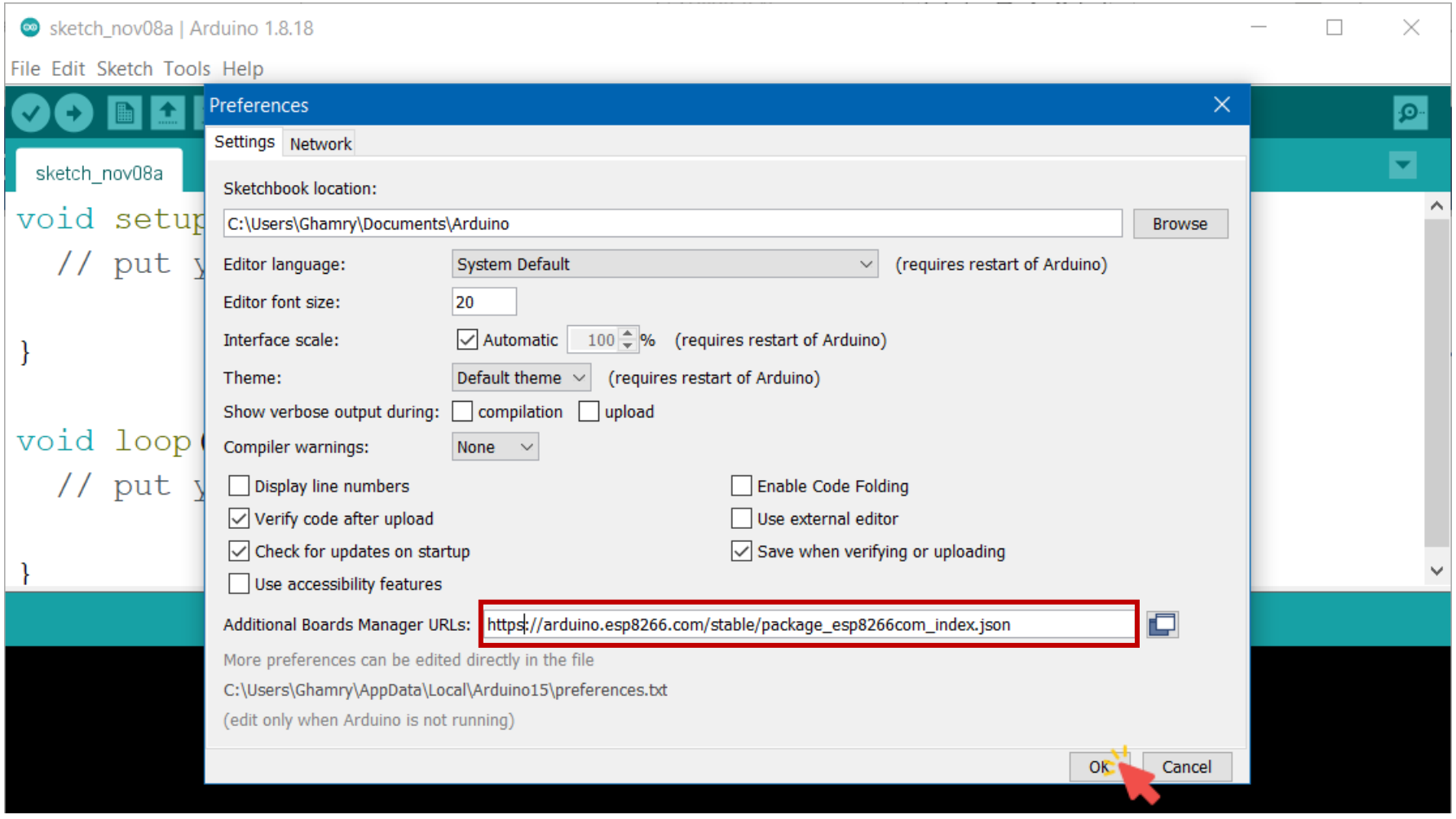
- First, open the Arduino IDE and go to File → Preferences.



Installing NodeMCU ESP8266 in Arduino IDE

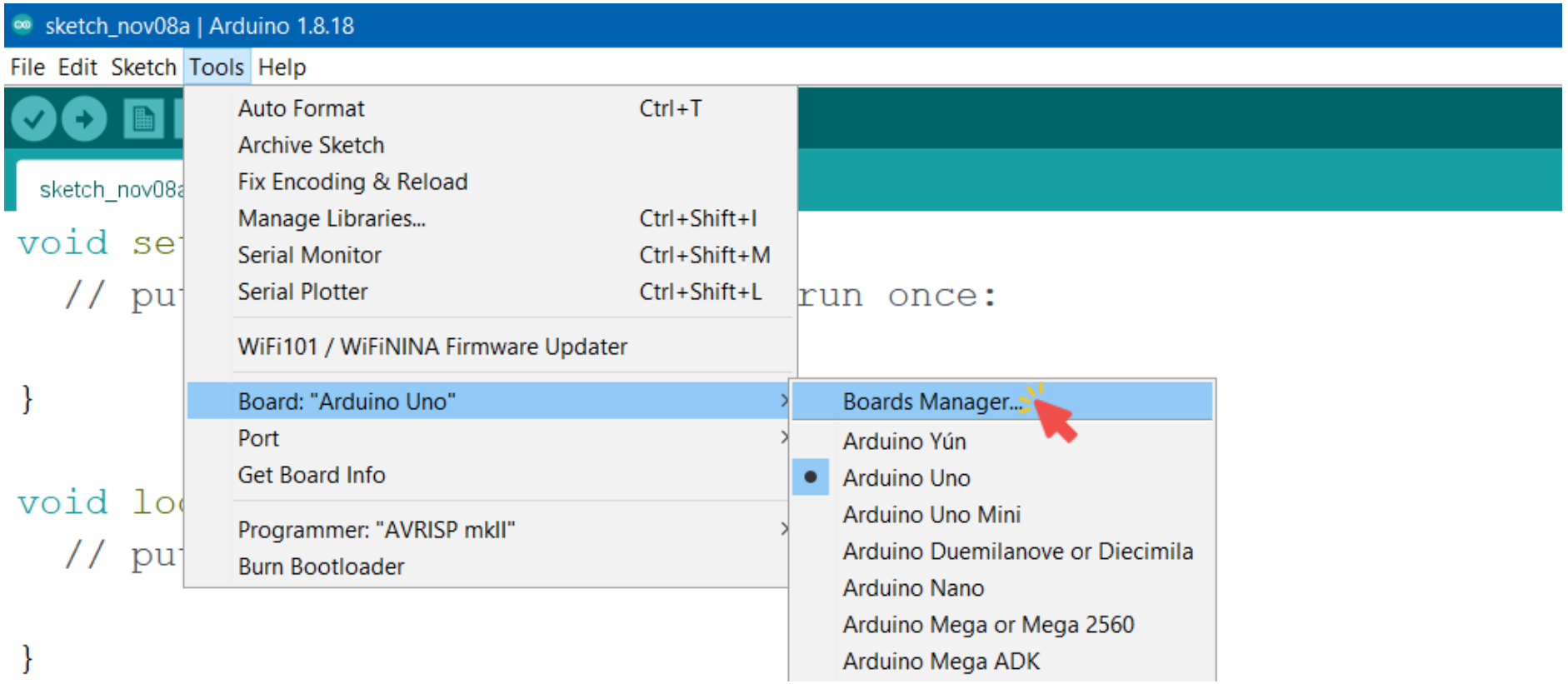
- On the Additional Boards Manager URLs, paste the following URL.

  https://arduino.esp8266.com/stable/package_esp8266com_index.json



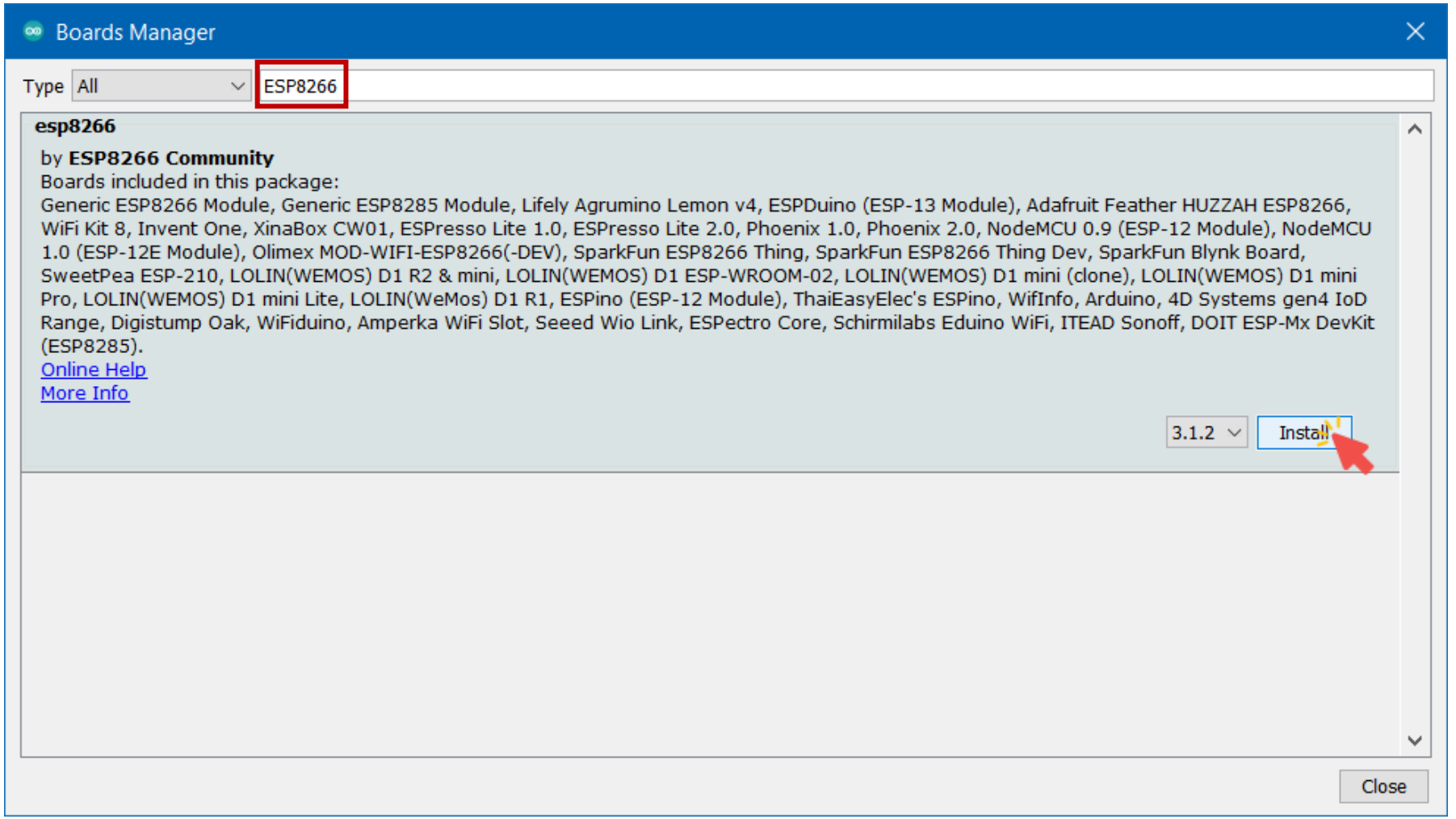
Installing NodeMCU ESP8266 in Arduino IDE

- Now go to Tools → Board → Boards Manager.



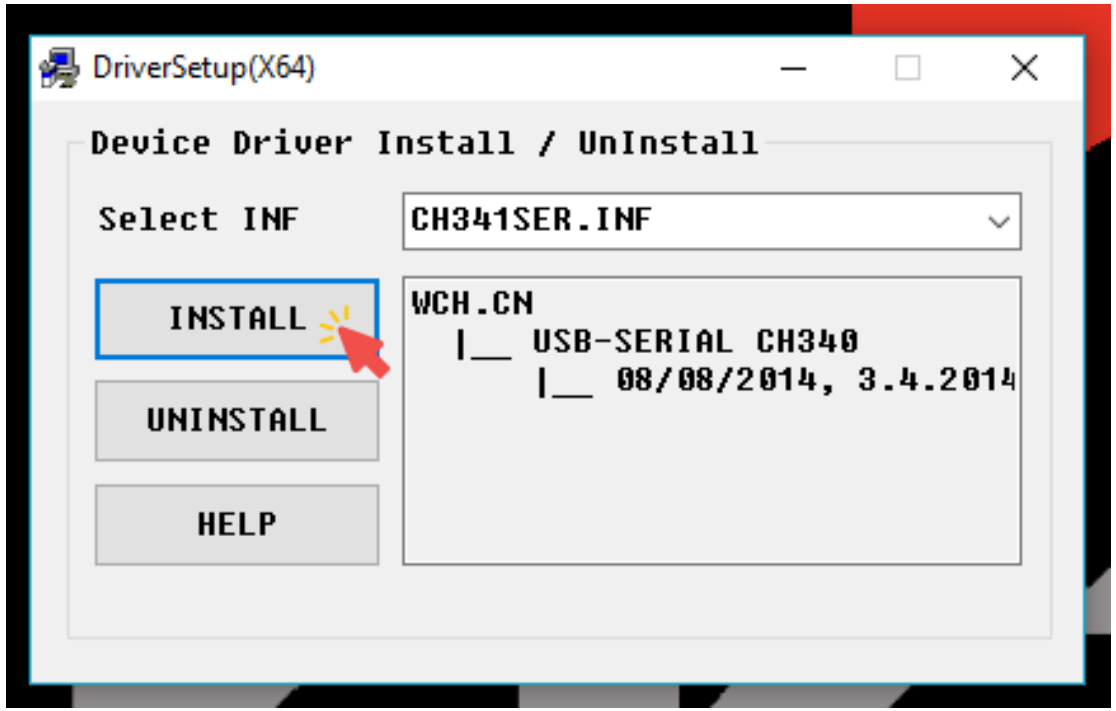
Installing NodeMCU ESP8266 in Arduino IDE

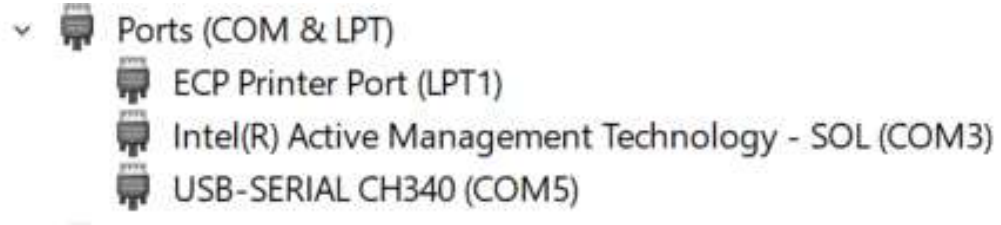
- Search ESP8266, and install esp8266 by ESP8266 community.



Installing NodeMCU CH340 Driver

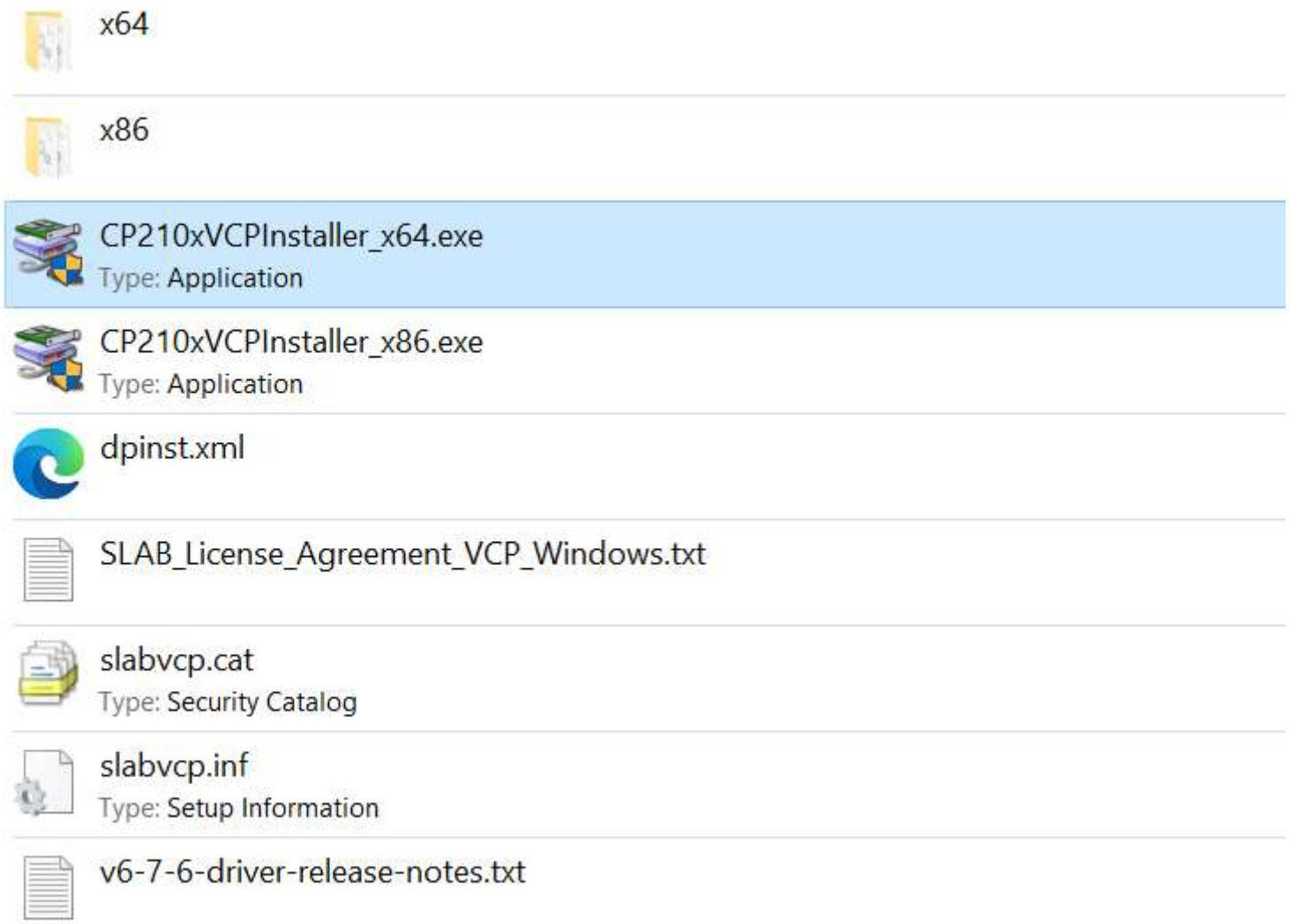
- Now, connect your NodeMCU ESP8266, and install the CH340 Driver.





Installing NodeMCU CP2102 Driver

- Install the CP2102 Driver.



Sketches

- A sketch is the name that Arduino uses for a program.

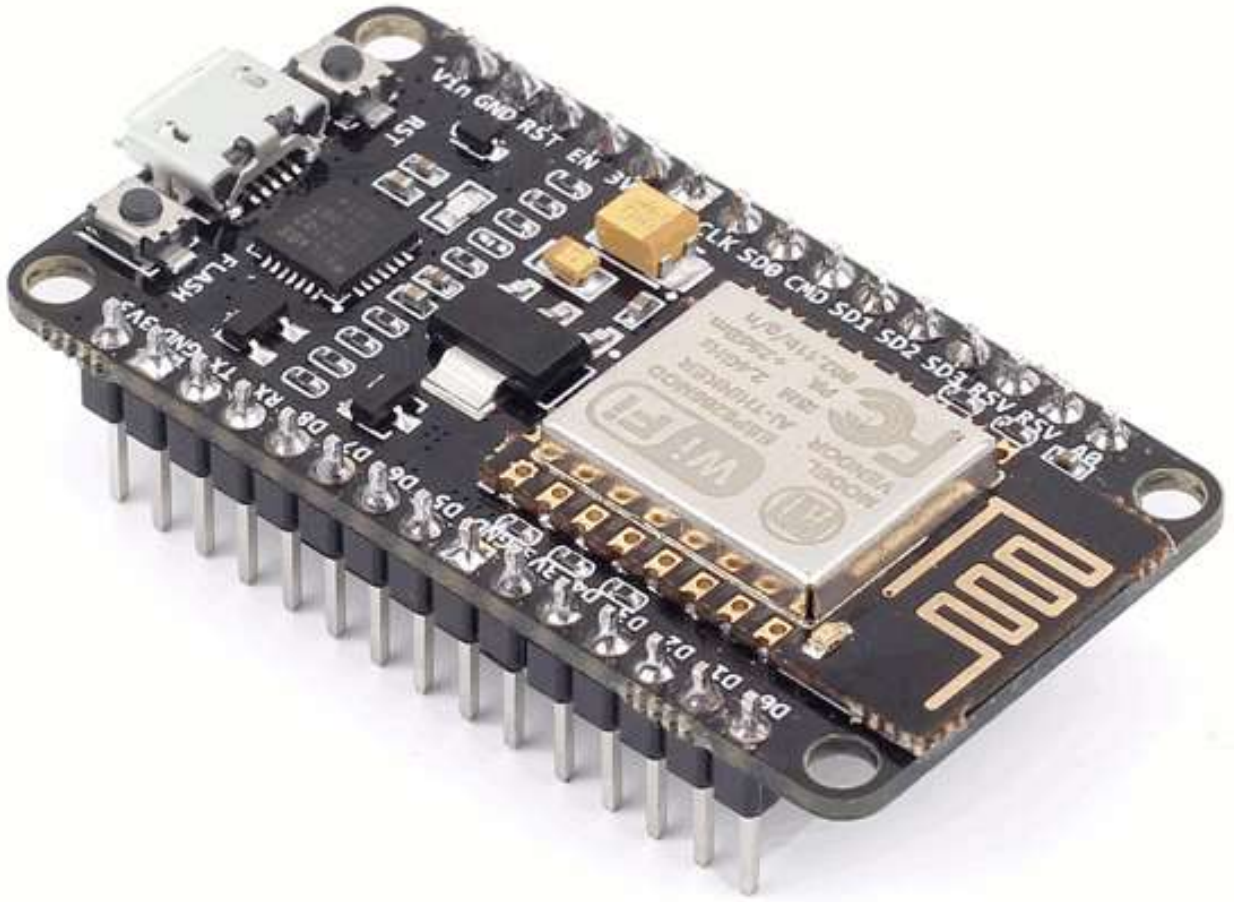
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Sketches

- There are two special functions that are a part of every Arduino sketch: `setup()` and `loop()`.
- The `setup()` is called once, when the sketch starts.
- It's a good place to do setup tasks like setting pin modes.
- The `loop()` function is called over and over and is heart of most sketches.
- You need to include both functions in your sketch, even if you don't need them for anything.

Your First NodeMCU Project: Blinking the On-board LED

- Turn the on-board LED on and off every second.



Your First NodeMCU Project: GPIO Code

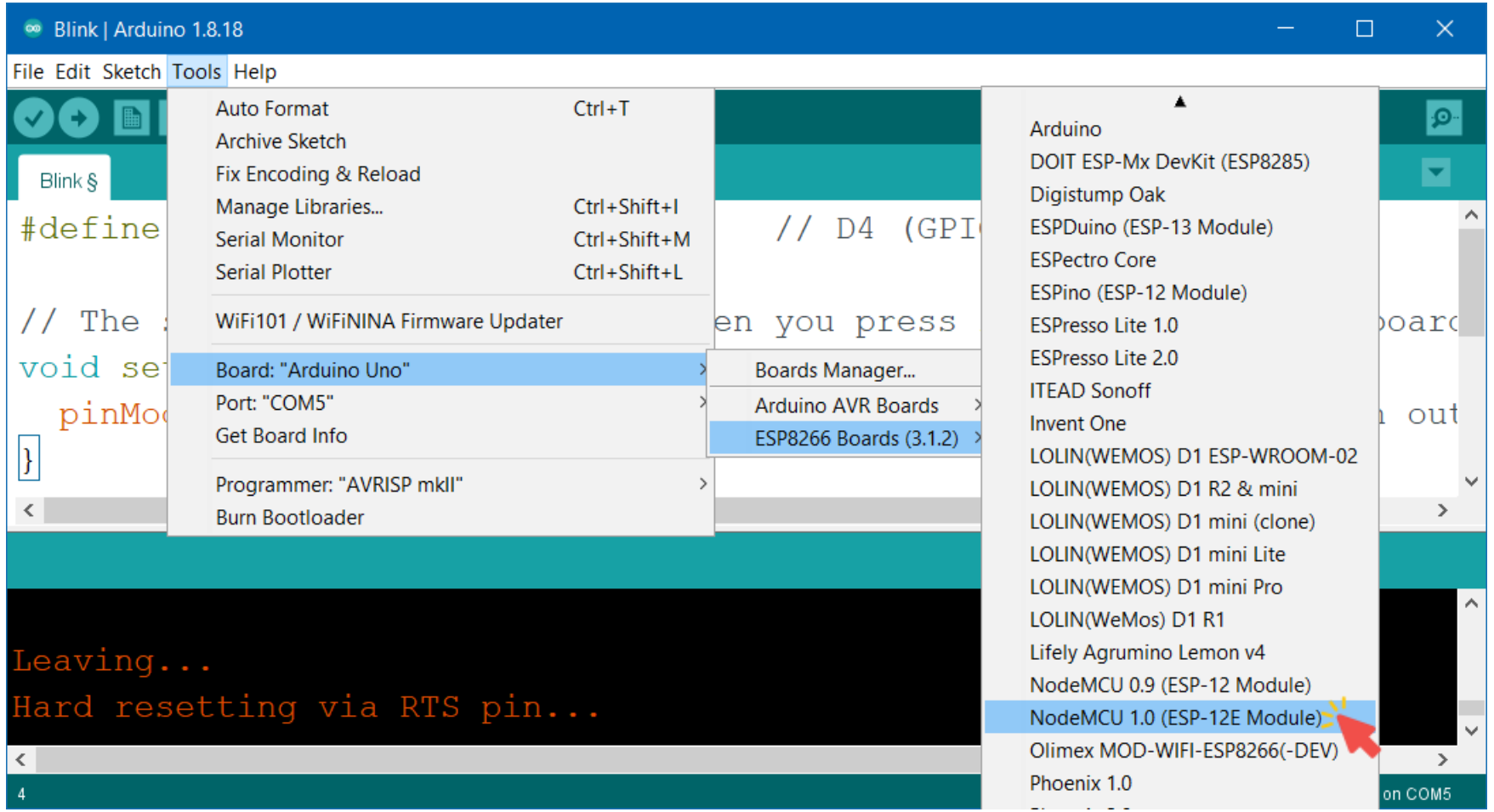
```
#define LED_PIN 2                       // D4 (GPIO2)

// The setup function runs once when you press reset or power the board
void setup() {
  pinMode(LED_PIN, OUTPUT);             // Initialize the pin D4 as an output
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);    // Turn the LED off
  delay(1000);                        // Wait for a second
  digitalWrite(LED_BUILTIN, LOW);     // Turn the LED on
  delay(1000);                        // Wait for a second
}
```

Your First NodeMCU Project: BCM Code

```
#define LED_PIN D4                    // D4 (GPIO2)

// The setup function runs once when you press reset or power the board
void setup() {
  pinMode(LED_PIN, OUTPUT);           // Initialize the pin D4 as an output
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);    // Turn the LED off
  delay(1000);                        // Wait for a second
  digitalWrite(LED_BUILTIN, LOW);     // Turn the LED on
  delay(1000);                        // Wait for a second
}
```

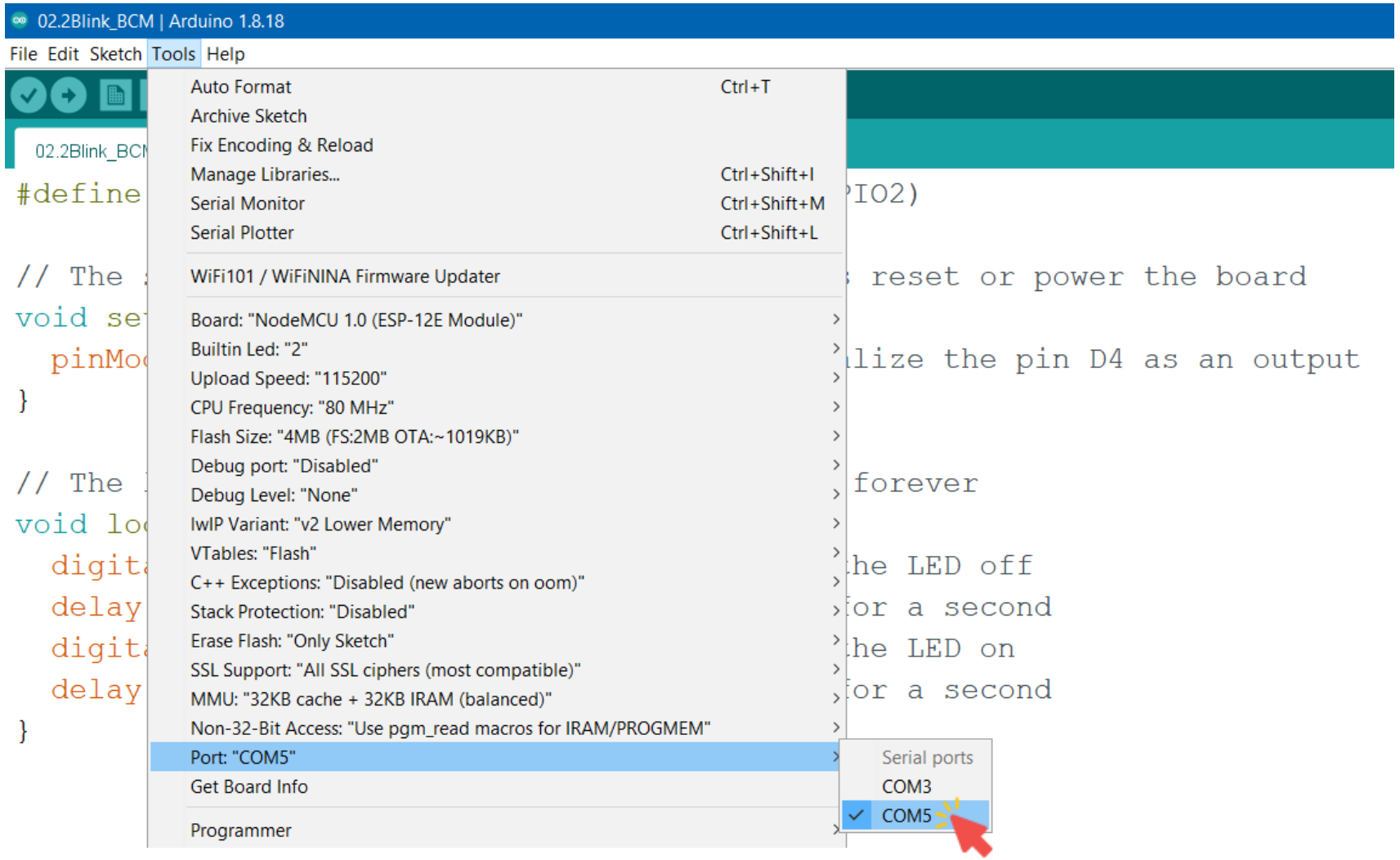
Your First NodeMCU Project: Arduino AVR Boards

- Now go to Tools → Board → ESP8266 Boards.
- Select NodeMCU 1.0 (ESP-12E Module).



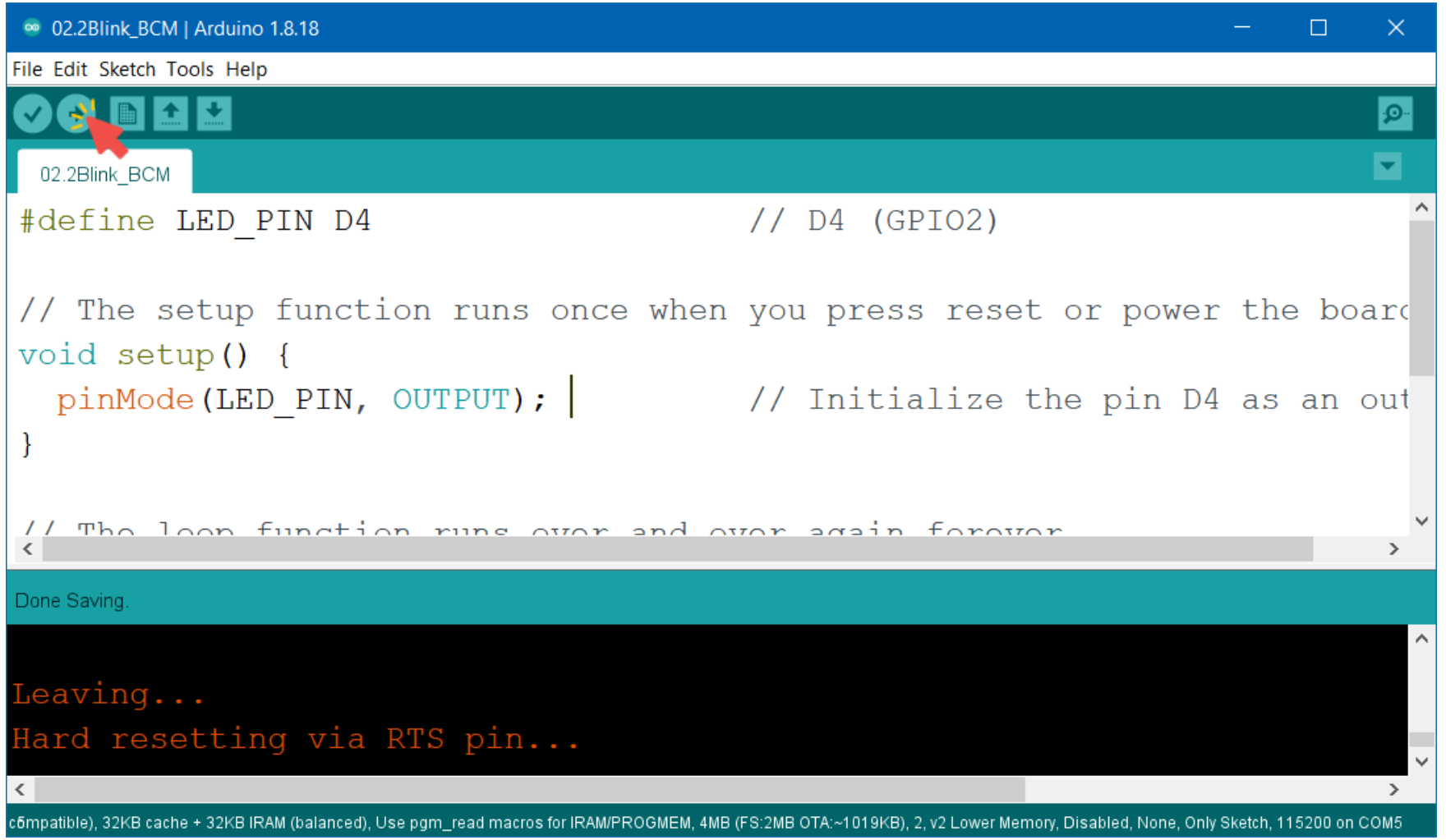
Your First NodeMCU Project: Port

- Go to Tools → Port, and select the port of your NodeMCU board.

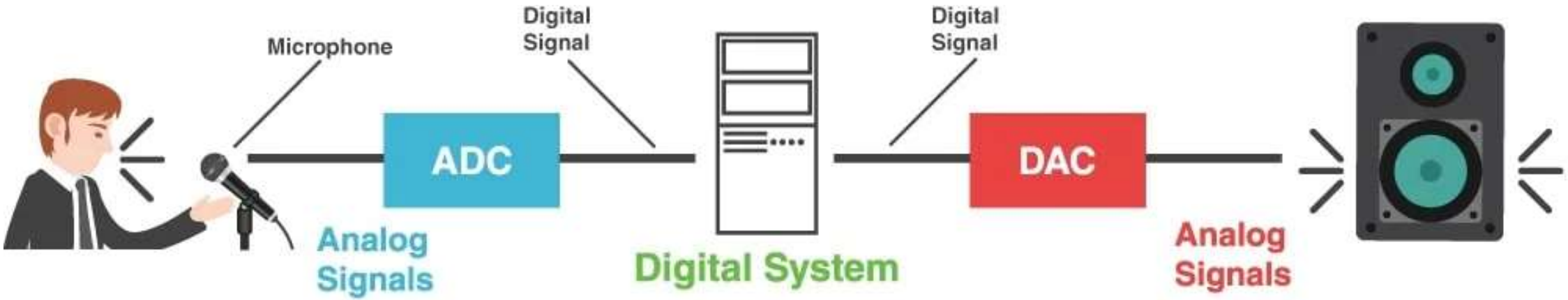


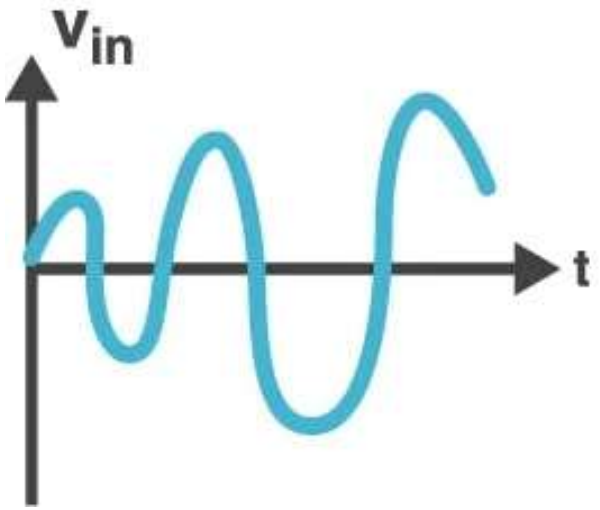
Your First NodeMCU Project: Upload a Sketch

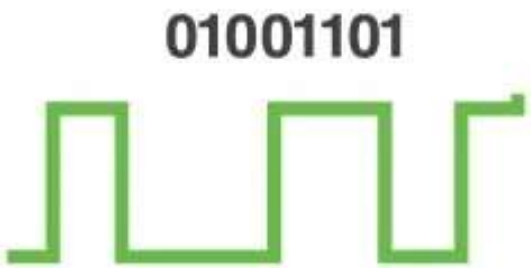
- Click the Upload button to program your NodeMCU with the sketch.

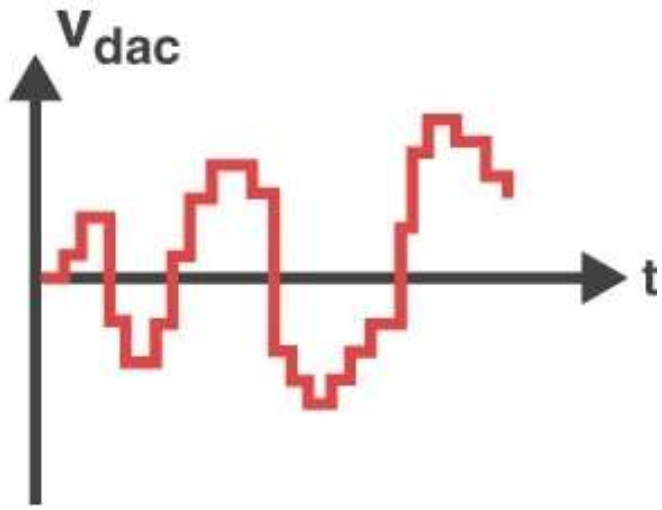


Analog vs. Digital Signals









Reading Analog Voltage

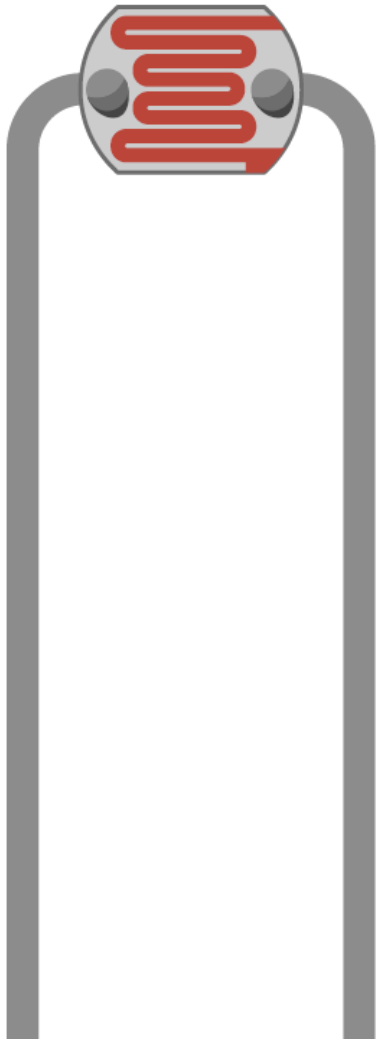
- The ESP8266-12E module has one ADC pin.
- The ADC pin has a 10-bit resolution, which means you'll get values between 0 and 1023.

```
value = analogRead(A0);
```

0	0	0	0	0	0	0	0	0	0	0V
1	1	1	1	1	1	1	1	1	1	3.3V

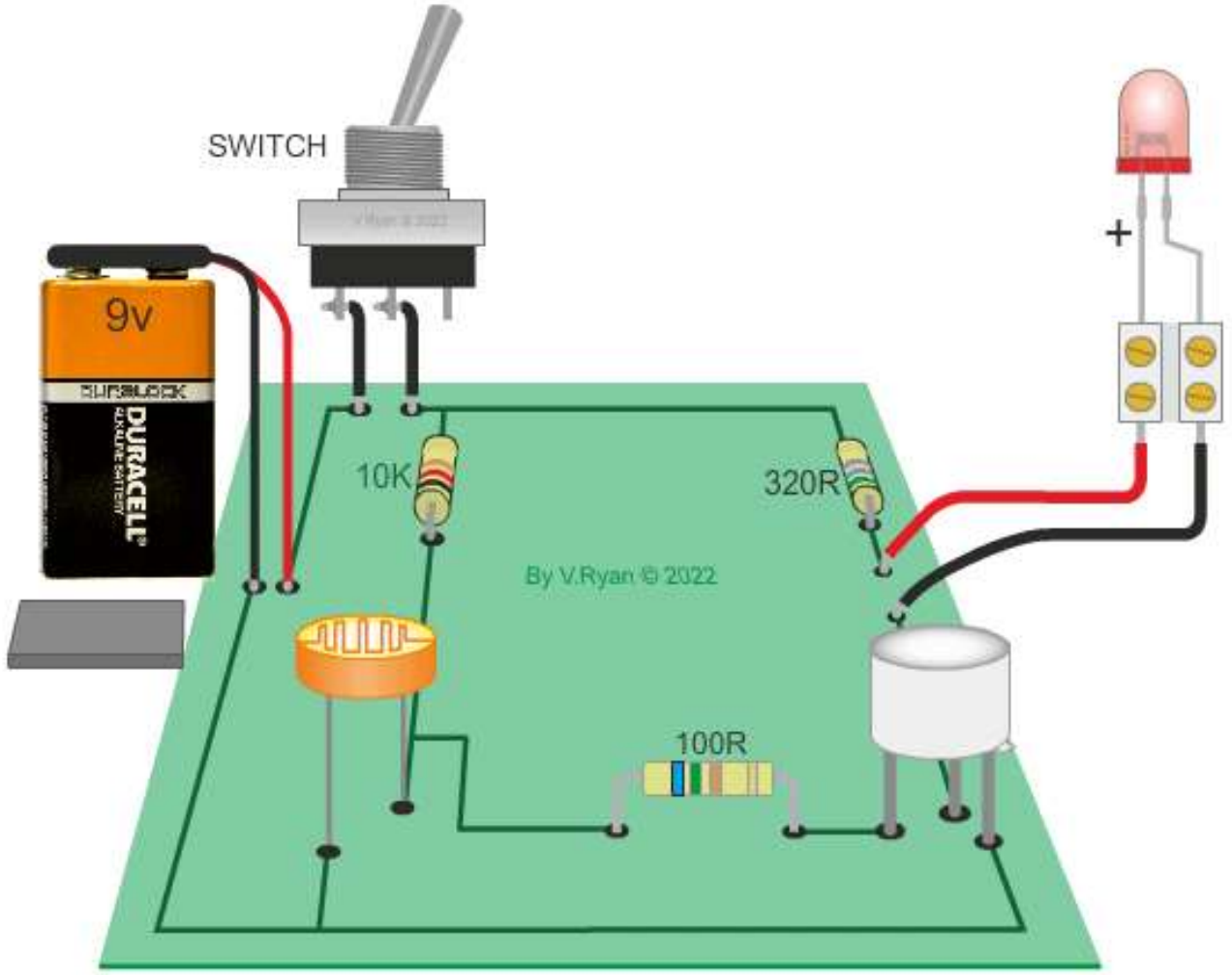
Photoresistor (Light Sensor)

- The photoresistor is a lightsensitive, variable resistor.

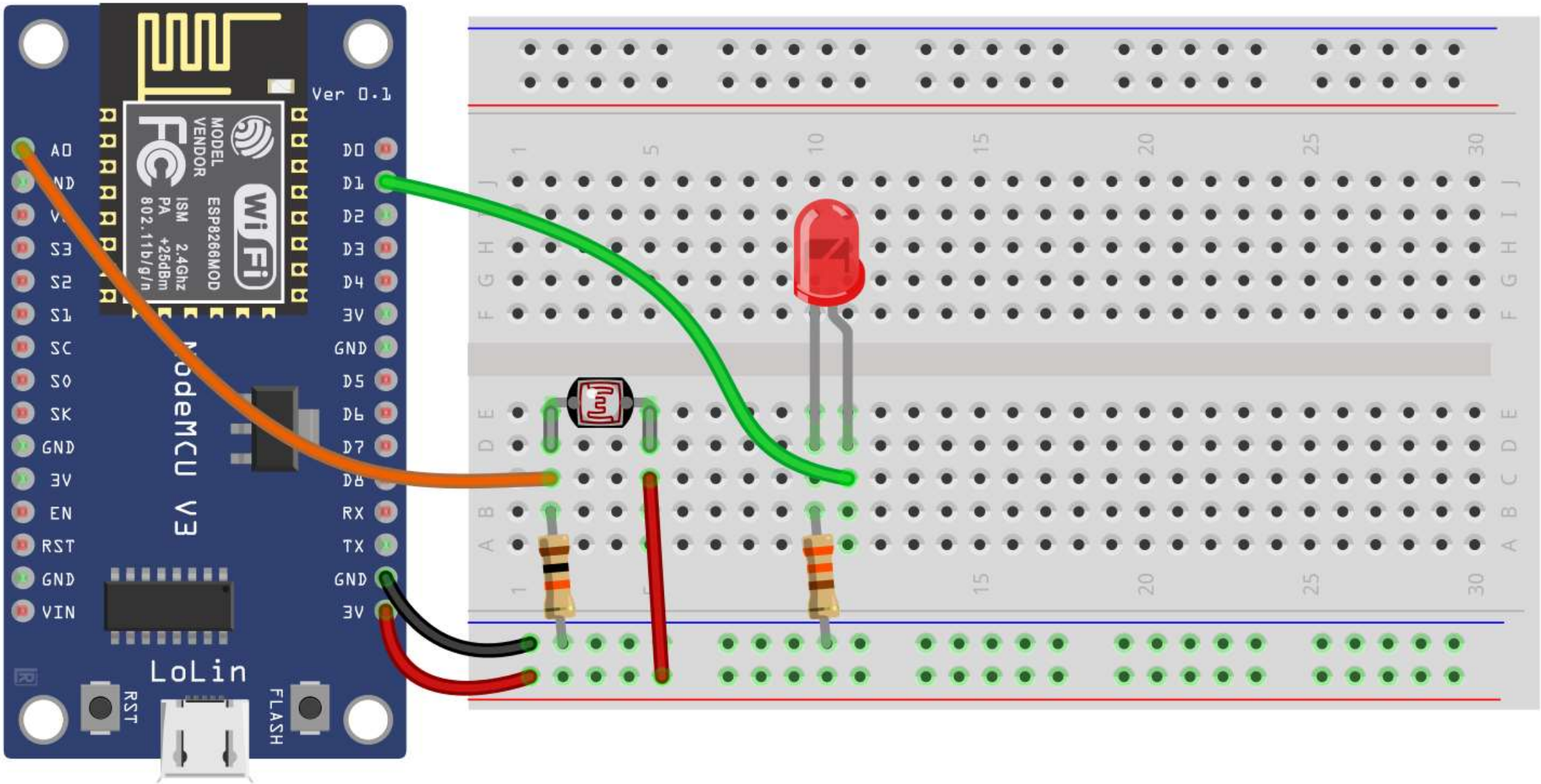


Photoresistor (Light Sensor)

- Photoresistors are perfect for making light-controlled switches.



Photoresistor: Circuit



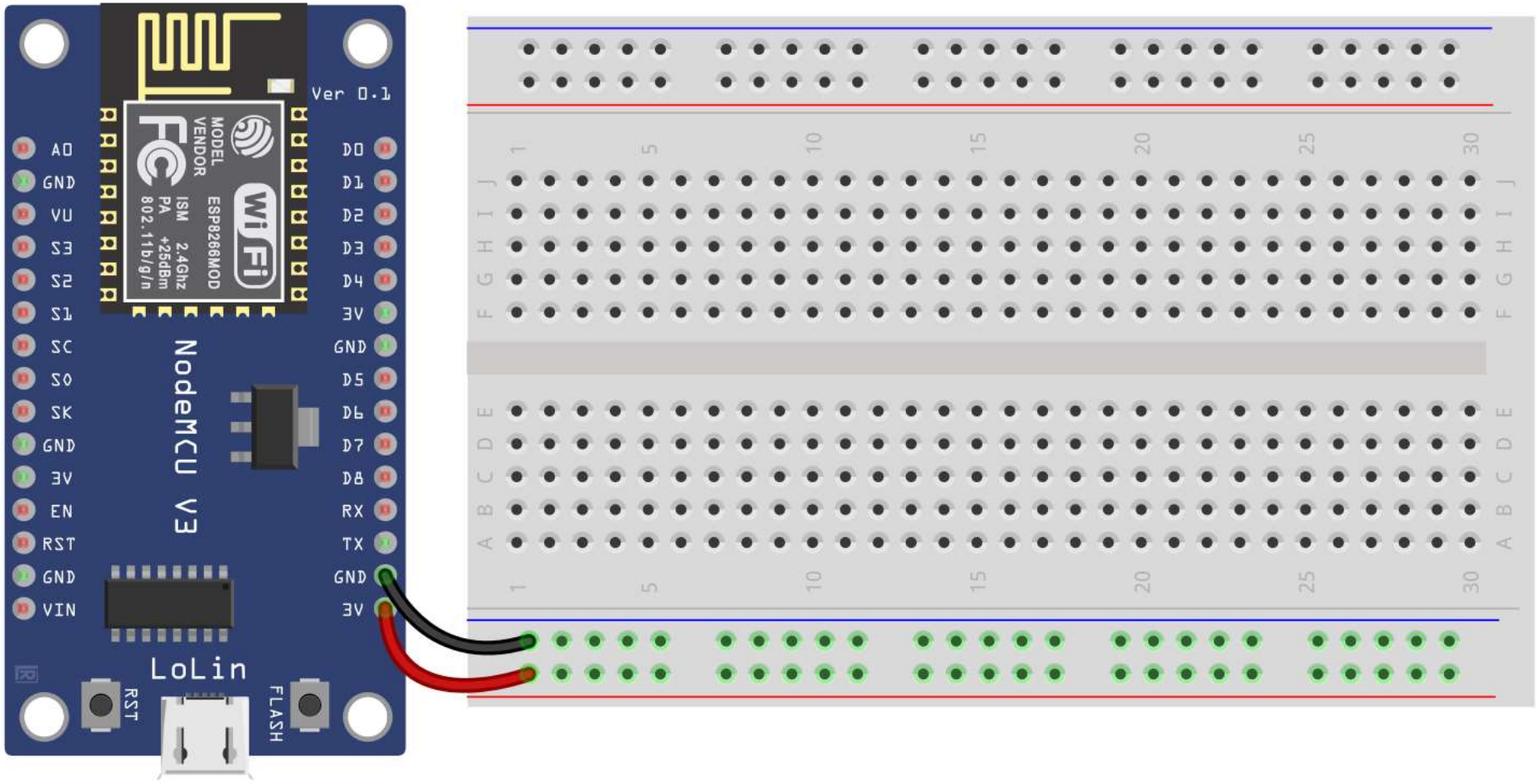
Photoresistor: Components

Now, we need the following components

- NodeMCU
- LED
- Photoresistor
- 330Ω Resistor
- 10KΩ Resistor
- Jumpers
- Breadboard

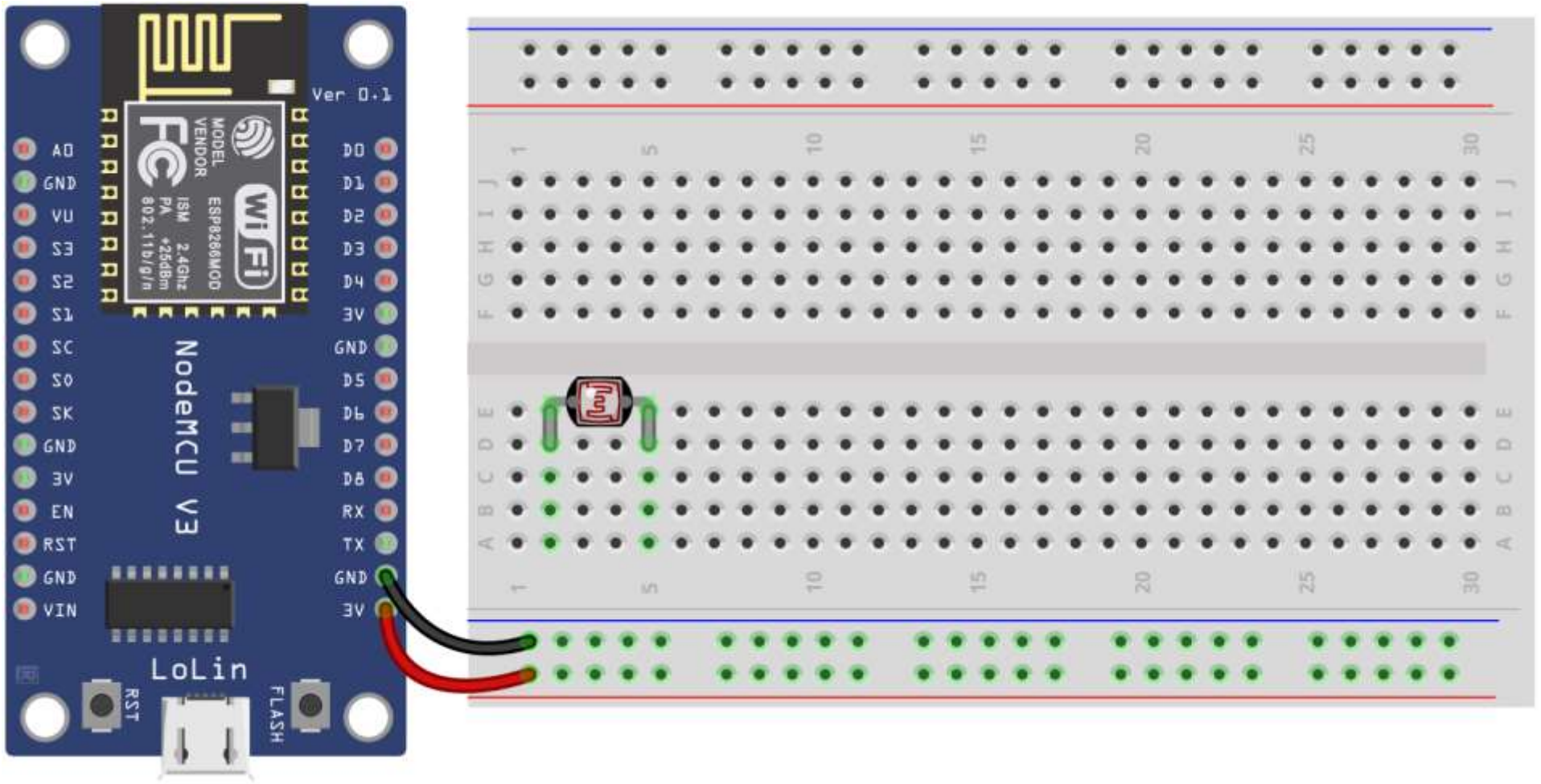
Photoresistor: Steps

1. Connect breadboard power (+) and ground (-) rails to NodeMCU 3.3V and GND, respectively.



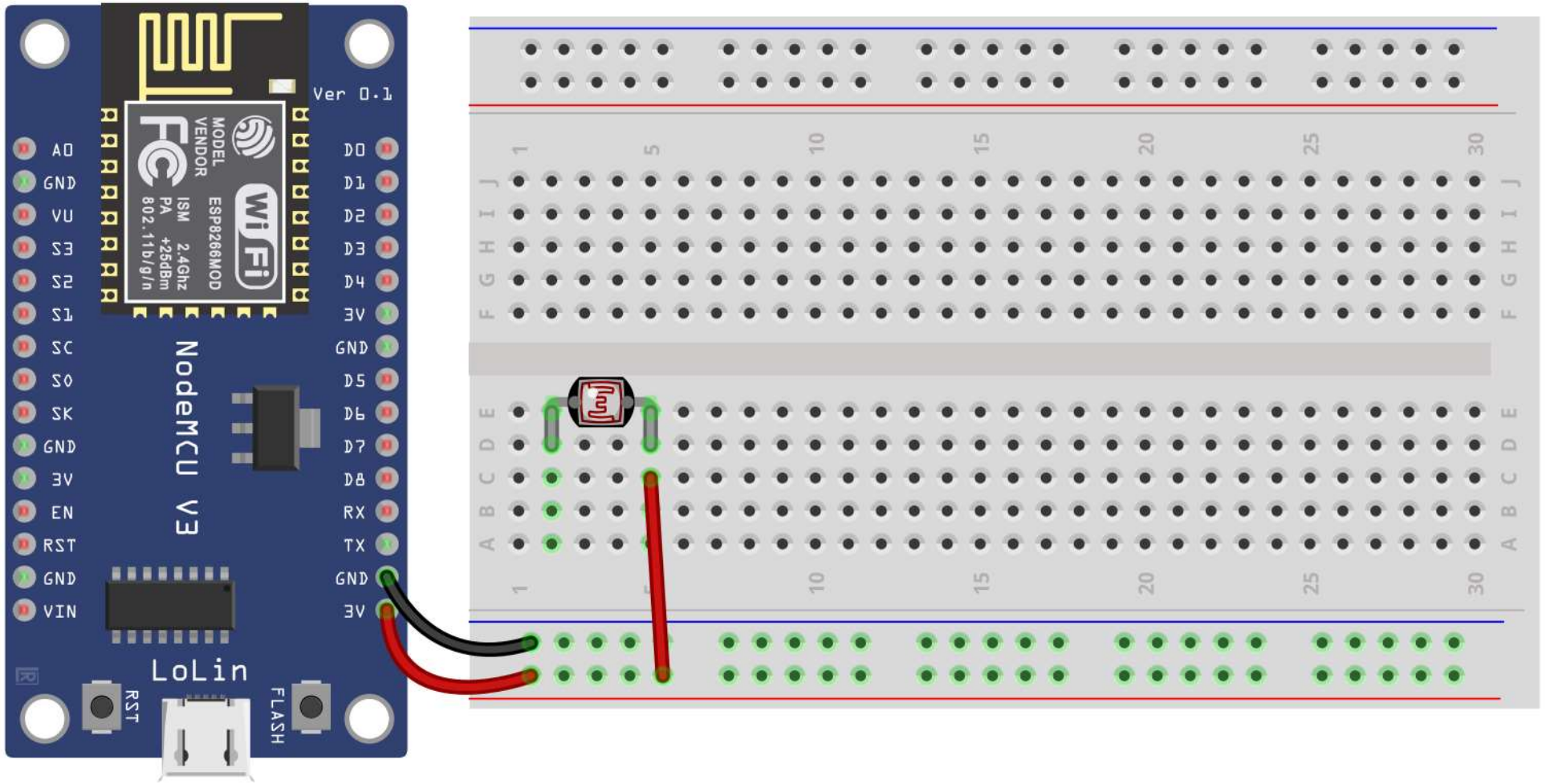
Photoresistor: Steps

2. Drag a photoresistor to your breadboard, so its legs plug into two different rows.



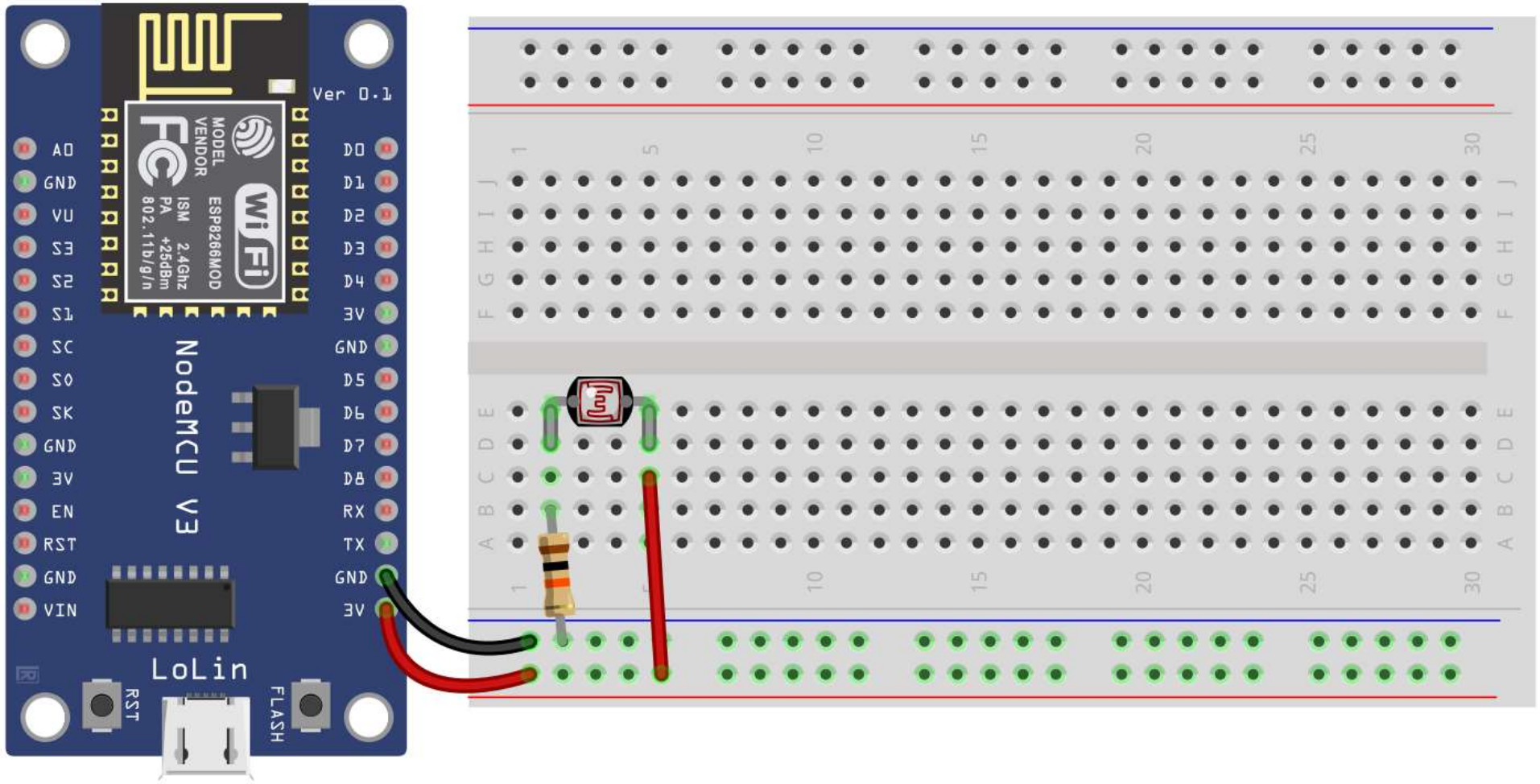
Photoresistor: Steps

3. Create a wire connecting one photoresistor leg to power.



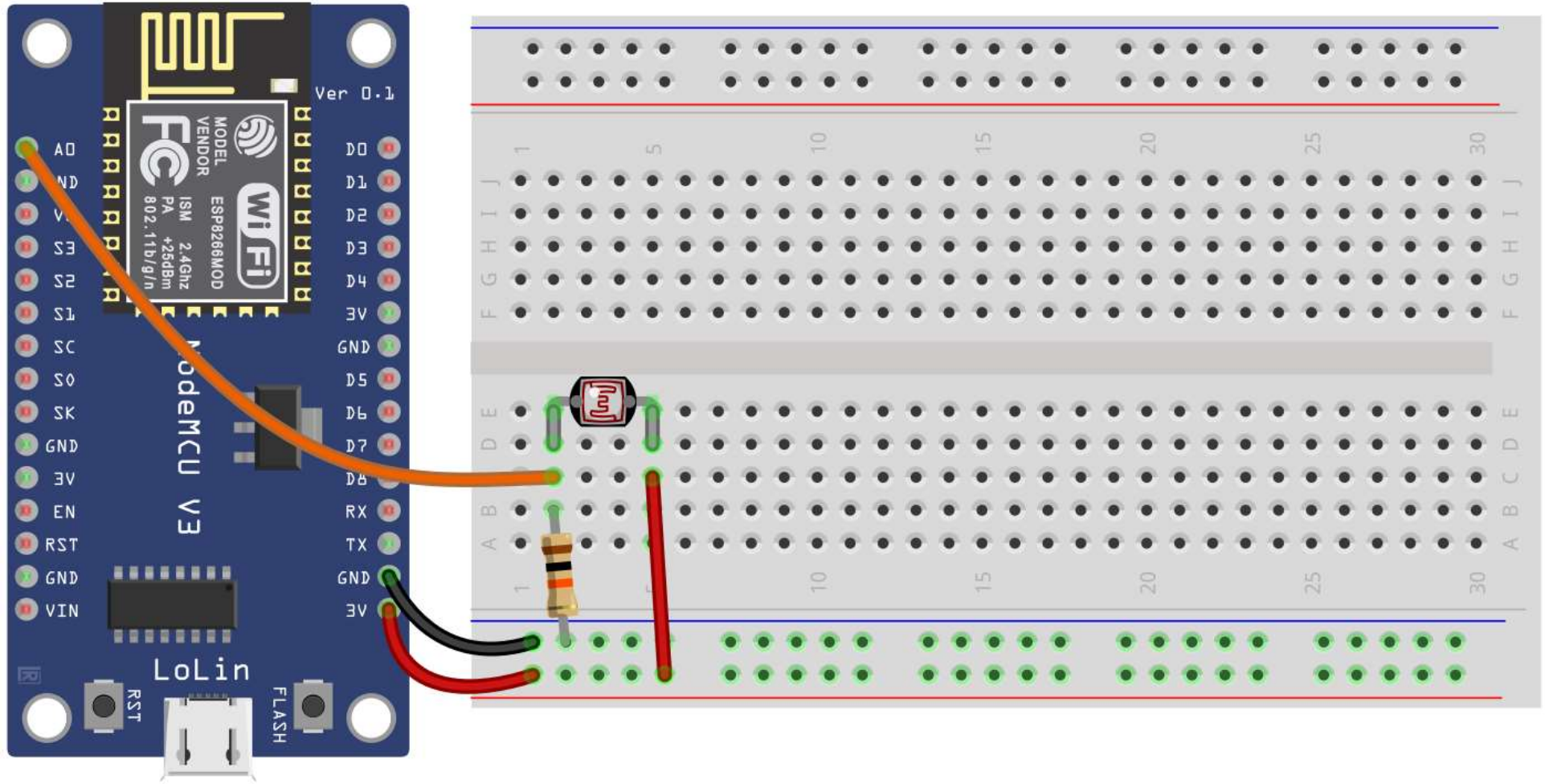
Photoresistor: Steps

4. Drag a 10KΩ resistor to connect other photoresistor leg to the ground.



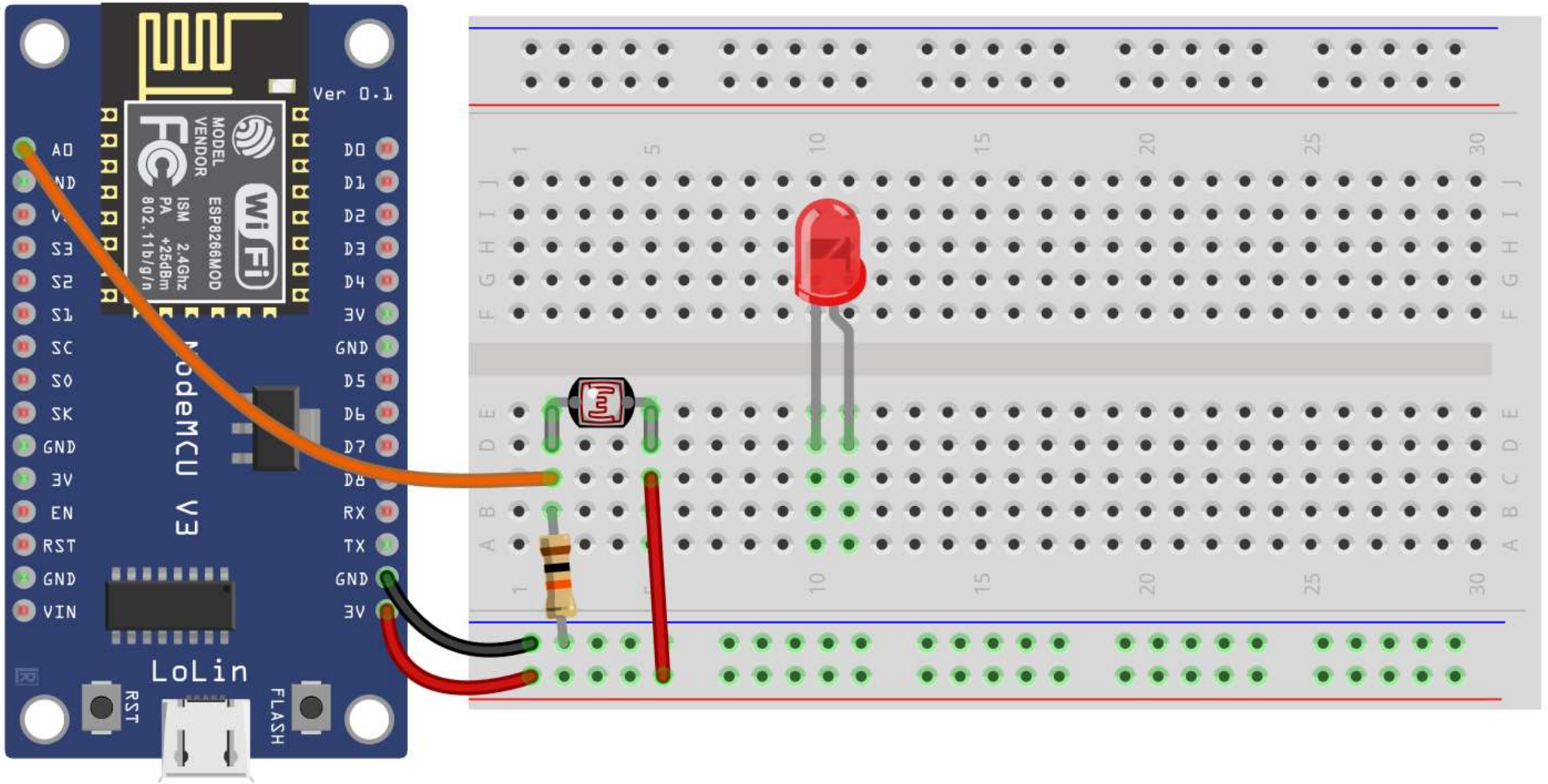
Photoresistor: Steps

5. Connect the photoresistor leg that is connected with the ground to the NodeMCU A0 pin.



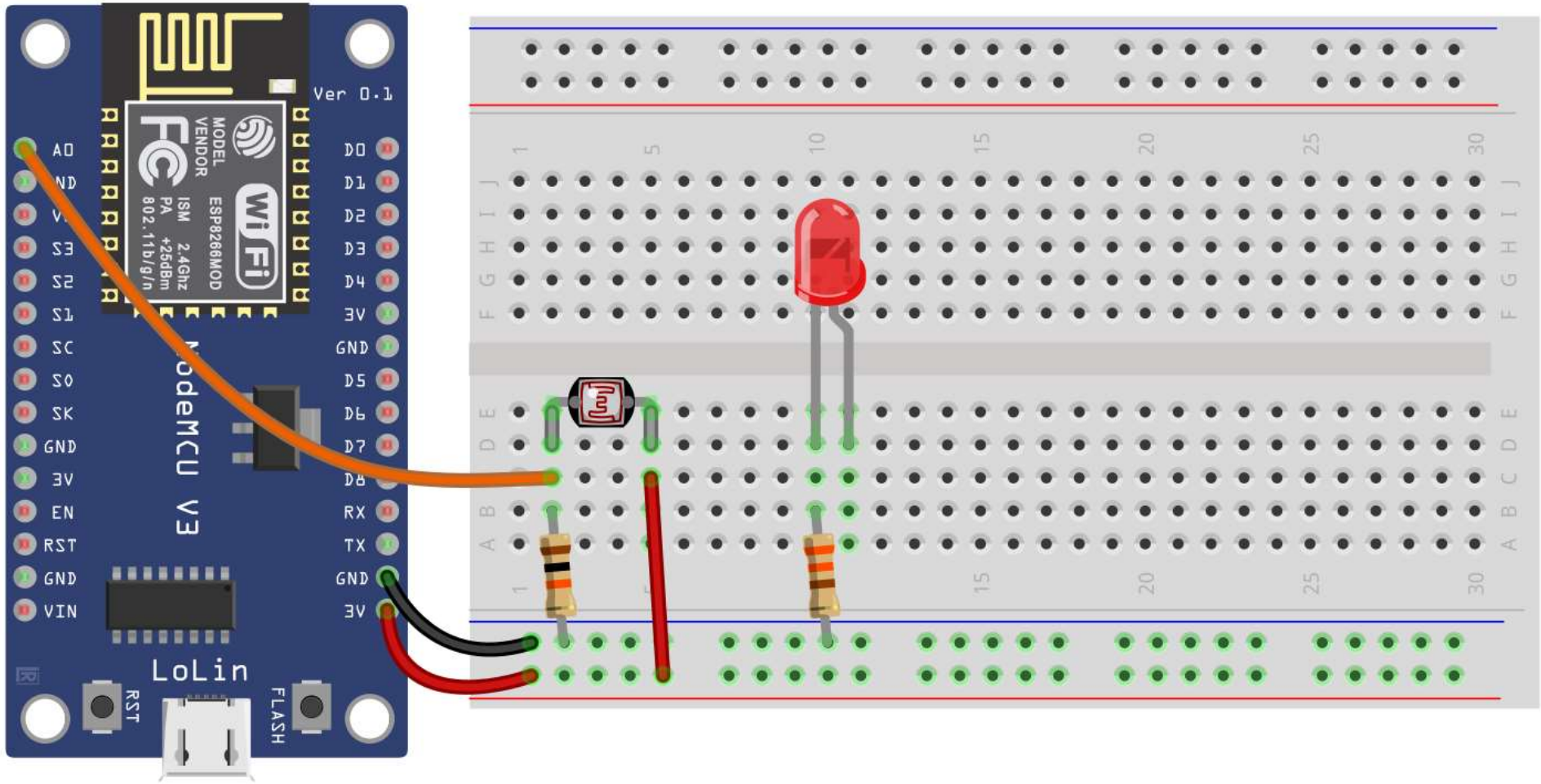
Photoresistor: Steps

6. Plug the LED into two different breadboard rows.



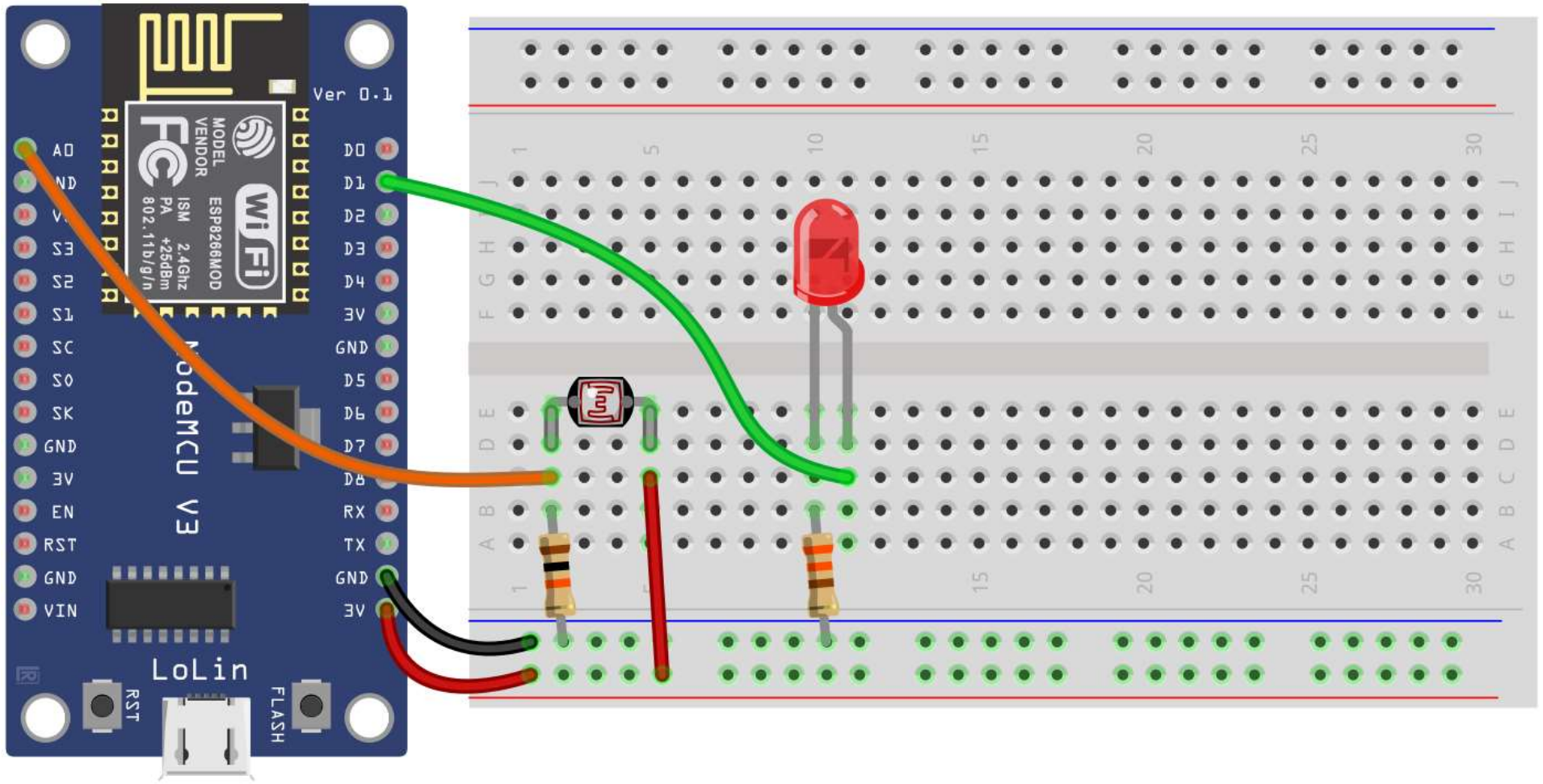
Photoresistor: Steps

7. The cathode (shorter leg) connects to one leg of a resistor of 330Ω, and the other resistor leg to the ground.



Photoresistor: Steps

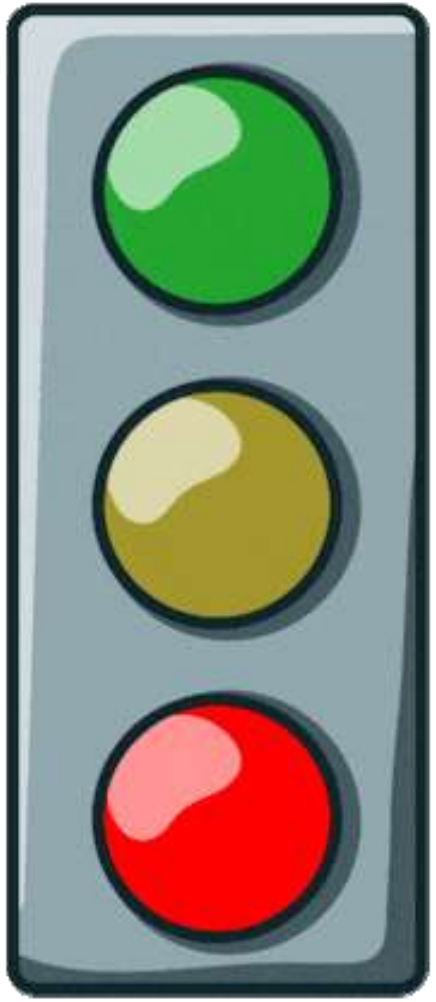
8. Wire up the LED anode (longer leg) to NodeMCU pin D1.



Photoresistor: Code

```
int photoresistor = 0;                  // Define a that holds a the photoresistor reading
int threshold = 750;                    // Define a threshold variable
#define LED_PIN D1                      // Define LED_PIN D1 (GPIO5)

void setup()
{
  Serial.begin(9600);                   // Start serial monitor
  pinMode(LED_PIN, OUTPUT);             // Set LED_PIN as an output pin
}

void loop()
{
  photoresistor = analogRead(A0);       // Read the brightness of the light
  Serial.println(photoresistor);        // Print the value of photoresistor on serial monitor

  if (photoresistor < threshold)        // If the photoresistor value is below the threshold
    digitalWrite(LED_PIN, HIGH);        // Turn on the LED
  else                                  // Else
    digitalWrite(LED_PIN, LOW);         // Turn off the LED

  delay(100);                           // Short delay
}
```

Assignment: Traffic Light



References and Tutorials

- IoT - NodeMCU (ESP8266) - Mohamed Yousef
- ESP8266 – Pinout
- A Complete Guide on ESP8266 WiFi Based Microcontroller
- NodeMCU ESP8266 Detailed Review
- NodeMCU ESP8266 - Arduino Store
- Introduction to NodeMCU V3
- ESP8266 Pinout Reference
- Get Started with Arduino IDE and ESP8266-NodeMCU
- How to Install CH340 Drivers
- NodeMCU PWM with Arduino IDE
- ESP8266 ADC – Read Analog Values